[illegible] 971124

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY *(Leave blank)*	2. REPORT DATE 11/21/97	3. REPORT TYPE AND DATES COVERED Final Technical Report 01 July 94 - 30 June 97

4. TITLE AND SUBTITLE AASERT-94 Broadband Signal Enhancement of Seismic Array Data: Application to Long-Period Surface Waves & High Frequency Wavefields	5. FUNDING NUMBERS Contract # F49620-94-1-0413
6. AUTHOR(S) Frank L. Vernon	
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) The Regents of the University of California, Scripps Institution of Oceanography, IGPP 0225, 9500 Gilman Drive, La Jolla, California 92093-0225	8. PERFORMING ORGANIZATION REPORT NUMBER AFOSR-TR-97-0696
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) AFOSR, 110 Duncan Avenue Room B115, Bolling AFB DC 20332-8080, Prgram Manager: Dr. Stanley K. Dickinson/NM NL	10. SPONSORING /MONITORING AGENCY REPORT NUMBER
11. SUPPLEMENTARY NOTES	
12a. DISTRIBUTION/AVAILABILITY STATEMENT Approved for public release, distribution unlimited	12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*

The research training activities started by identifying and picking seismic events from various broadband arrays and networks. The next step was to associate these events with existing catalog locations to build databases of local, regional and teleseismic events which are being used in this research project as well as other seismic verification projects at our institute and also other universities. The data processing aspects included learning to use the CSS 3.0 database format to allow the management of datasets with thousands of events. The seismic processing included identification of local, regional, and teleseismic phases which are essential for seismic discrimination research. This involved learning the TCL/TK language and the application of this knowledge to provide a graphical user interface to build CSS databases. This program is being developed to support field deployments of seismic stations which will ensure the proper documentation of instrument responses, site locations, and calibration parameters. In addition, this software has been used to support deployments of the Wyoming portable broadband seismic and infrasonic experiments, and with all aspects of data assembly, data processing, data distribution to the research community.

14. SUBJECT TERMS	15. NUMBER OF PAGES 124
	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT Unclassified	18. SECURITY CLASSIFICATION OF THIS PAGE Unclassified	19. SECURITY CLASSIFICATION OF ABSTRACT Unclassified	20. LIMITATION OF ABSTRACT SAR

NSN 7540-01-280-5500

Standard Form 298 (Rev 2-89)
Prescribed by ANSI Std. 239-18
298-102

DTIC QUALITY INSPECTED 3

19971217 003

This AASERT award was used for Aaron Geddins, a minority undergraduate student at the University of California at San Diego. The AASERT research project was in the Institute of Geophysics and Planetary Physics at the Scripps Institution of Oceanography. As a part of this program, Aaron was required to research areas of local, regional and teleseismic seismology. In this process, it was necessary to gain a thorough knowledge of computer hardware and software, data management, and office administration. Aaron has provided several presentations on his various aspects of his work to IGPP groups.

Aaron's initial task was to spend part of his first year picking arrivals from seismic events following the Mw=6.8 Northridge earthquake of 17 January 1994. In order to do this, he was trained on a program called dbpick. Dbpick allows the user to see the digital display of the waveforms and estimate where the arrivals are. It's the analyst's job to give the precise location of these arrivals and give its amplitude and range. With the collection of this data, Aaron calculated the locations and magnitudes of the seismic events.

After the completion of the Northridge data analysis, Aaron was tasked with learning about what is involved in deploying a seismic station. He learned about the different ways to orient a seismometer, what a datalogger was and how it is set up, how to assemble a solar panel and how to connect it with the datalogger and seismometer, how to use a compass in aligning a solar panel, and how to secure a station from possible vandalism. Knowing the proper way to install a station is vital in order to record seismic events accurately.

With this knowledge of station deployment, Aaron was instructed on how to provide proper documentation for each deployment. This included the preparation of station parameter files which provide a time history of each station's operating parameters. A station parameter file (stapar) is a data file that consists of information such as: latitude, longitude, elevation, sensor serial number, datalogger serial number, sample rate, and other technical aspects that define the seismic station. The stapar file is necessary in order to keep account of the current condition of the seismic stations. If a component of the station such as the seismometer was replaced, a new stapar file would be created to indicate that change. Another example would be if the researcher would send a technician to the station to change the sample rate. Thus, a new stapar file would be created to show the new values for the fields changed.

After being well versed in picking of arrivals, station deployment, and creating and understanding stapar files, Aaron had the necessary tools to write a computer program named field2db. Field2db is a graphical user interface application written in tcl/tk and is intended to provide a simple interface to input data supplied by field technicians to create or update the appropriate tables in the CSS3.0 database format. Field2db has three main functions, to view stapar files in a much more readable format, create stapar files and database tables from scratch (meaning from a new seismic network), and edit existing database tables. In the past, the best way to view a stapar was on a printout. Even then, there are so many fields that sometimes one can lose track of what information that was in question. Field2db will view a stapar with only selected fields showing. If the user wants to see more fields or specific fields, there is a menu button 'VIEW' that will allow them to do just that. If the user wants to see all the fields in a stapar file, then field2db will supply a horizontal scrollbar in order to view the 'hidden' fields.

The creation of stapar files and database tables is a little more involved process. In this aspect of field2db, the user is asked to fill in information about the seismic network and the stations that make up that network. It has been discovered that most of the data needed to create stapar files and database tables can be produced from logfiles. Logfiles are files that are produced automatically and require no human interference. With this new found discovery, it makes for less data entry for the human user and thus allows for less mistakes in the stapar files and database tables. This feature of field2db is a window-based, menu-driven feature, thus the user will have all the information needed in order to enter the data. A text window drives the user through the necessary steps and highlights the proper buttons to be clicked. Once the user has entered the proper information, the Build Database button will be highlighted. Once this button is clicked, the database tables will be written. These tables are written in CSS3.0 format and can be viewed by using another program called dbe. The database tables are written by using the datascope utilities which have been developed at the Joint Seismic Program Center at the University of Colorado. Along with logfiles, parameter files are also used to cut down on the amount of entries that a user has to make. For instance, for a given REFTEK datalogger, certain sample rates correspond to certain filters that are to be used. So a user should never have to enter the filters needed to get a certain sample rate to 40 samples per second. This information should be put into a parameter file. The net result is a simple interface which either portable deployments or regional networks can use to generate CSS3.0 database for archiving, data analysis, or data distribution.

```
#!/bin/sh
# This comment extends to the next line for tcl \
exec dbwish -f $0 $*
#
#
#



auto_load tkerror
########################################
#Procedures to find the correct network

proc net_search { is_net } {
   global net_path net_name pattern
   set net_name $is_net
   set path [exec sh -c "ls $net_path"]
   foreach network $net_path {
        if {$net_name == "northridge"} {
           northridge_out $network
           search entry
        } elseif {$net_name == "OTHER"} {
             other_search
        } else {
             set pattern $network/$net_name/stapars
             search entry
        }
   }

}

proc create_table {} {
   global ep year date epoch_clk time key new_net t \
          rf_sta rt minus tabletime i j ok yr jday hr min sec yrday

   set length [string length $t($time)]
   set length [expr ($length - 2)]
   set rf_sta -

#puts "time is $t($time)"
#puts "what length:  $length"
#set newclock [str2epoch "$yr$jday$hr$min.$sec"]

   set ok($i,$j,$time) 0
   set yrday $yr
   set string [string trimleft [string range $t($time) 2 $length]]
   set newclock $string
   set cat "S"
   set t($time) "$newclock$cat"
   set rt($time) "$newclock$cat"
   set yr [string trimleft [string range $newclock 0 1]]
   set jday [string trimleft [string range $newclock 2 4]]
   set hr [string trimleft [string range $newclock 5 6]]
   set min [string trimleft [string range $newclock 7 8]]
   set sec [string trimleft [string range $newclock 10 11]]

puts "this is create_table"

   set tabletime($time) [str2epoch "$yr:$jday:$hr:$min:$sec"]
   set minus($time) [expr ($tabletime($time) - .1)]
   set tableminus($time) [yearday "$minus($time)"]
   set minustime [strtime "$minus($time)"]
   set tableyrday($time) [yearday "$tabletime($time)"]
   set tabletyme($time) [strtime "$tabletime($time)"]
```

```
      set ep [epoch "$yrday$jday"]
      set year [yearday "$ep"]
      set date [strtime "$ep"]
      set stdate [strdate "$ep"]
      set epoch_clk($time) "($year) $date"
      set epoch_clk($time) "($tableyrday($time)) $tabletyme($time)"
      set epch($time) "($tableminus($time)) $minustime"
      #puts "the date is:  $epoch_clk($time)"
      #puts "with the minus the time($t($time)) is : $epch($time)"

      mkdir $new_net
      mkdir $new_net/stapars
      exec sh -c "cp $t($time) $new_net/stapars"
      writetable
   }

   proc calc_band {rate freq} {
      global band
      set ch_code ""
      ################################################
      # calculate the band and channel codes
      ################################################
      set ck_freq [expr (1 / $freq)]
      set ck_samp $rate


      if {($ck_samp >= 80.0) && ($ck_freq \
         < 10.0)} {
         set band "e"
         set ch_code E
      } elseif {($ck_samp >= 10.0) && ($ck_samp < 80.0) \
         && ($ck_freq < 10.0)} {
         set band "s"
         set ch_code S
      } elseif {($ck_samp >= 80.0) && ($ck_freq \
         >= 10.0)} {
         set band "h"
         set ch_code H
      } elseif {($ck_samp >= 10.0) && ($ck_samp < 80.0) \
         && ($ck_freq >= 10.0)} {
         set band "b"
         set ch_code B
      } elseif {($ck_samp > 1.0) && ($ck_samp < 10.0)} {
         set band "m"
         set ch_code M
      } else {
         puts "no band code for the samprate and corner!"
      }
      return $band
   }

   proc other_search {} {
      global sitename stacode new_net netname pattern
      set z [uniqueW ""]
      toplevel $z
      set stacode 1
      set sitename(1) -
   #puts "we are in other!"
      label $z.netname \
           -text "enter network name: "
      entry $z.netentry \
           -width 25 -textvariable new_net

      label $z.netcode \
        -text "enter network description: "
   entry $z.entrycode \
        -width 50 -textvariable netname

   label $z.all_path \
        -text "give path name of stapar to open:  "

   entry $z.entry_path -width 35 -textvariable pattern
   #bind $z.entry_path  <Return> "search entry; destroy $z"

   button $z.dismiss -text "Dismiss" \
        -command "destroy $z"

   blt::table $z \
        $z.netname       1,1     -anchor w -fill x \
        $z.netentry      1,2     -anchor w \
        $z.netcode       2,1     -anchor w \
        $z.entrycode     2,2     -anchor w \
        $z.all_path      3,1     -anchor w \
        $z.entry_path    3,2     -anchor w \
        $z.dismiss       4,2     -anchor e -fill x

   focus $z.netentry
   bind $z.netentry <Return> "focus $z.entrycode"
   bind $z.netentry <Tab> " focus $z.entrycode"
   bind $z.entrycode <Return> "focus $z.entry_path"
   bind $z.entrycode <Tab> "focus $z.entry_path"
   #bind $z.entry_path <Return> "sta_info $z; search entry; destroy $z"
   #bind $z.entry_path <Tab> "sta_info $z; search entry; destroy $z"
}

proc FindDb { is_net } {
   global net_path net_name pattern1 network pattern
   set net_name $is_net
   set path [exec sh -c "ls $net_path"]
   foreach network $net_path {
        if {$net_name == "northridge"} {
                northridge_out $network
        } else {
                   set pattern1 $network/$net_name/css30/field
                   mkdir field
                   exec sh -c "cp $pattern1/*.* field"
                   set pattern field
        }
        EditDbWindow
        #GetStation
   }
}

proc CheckTimes {current} {
    global Fields N Sequence Savetype Checkbutton request
    global Ok wanted f name Possible stalist get_sta

    set title $name
    set w [uniqueW ""]
    toplevel $w
    wm title $w $title

    set maxrow 20
    set maxfields 10
    set lastrow [expr $maxrow+20]
    set span [expr $maxfields / $maxrow + 1]
    frame $w.f2
    button $w.f2.cancel \
        -text "cancel" \
        -command "set Ok 0 ; destroy $w"
    button $w.f2.ok \
        -text "ok" \
        -command "set Ok 1; time_span; destroy $w"
    pack $w.f2 -side bottom -fill x
    pack $w.f2.cancel $w.f2.ok -side left -fill x -expand yes

    set f $w.f3
    frame $f
    pack $f -side top -fill x

    if { [info exists Fields]} {
        unset Fields
        unset Sequence
        unset Checkbutton
    }

    set i 0
    set N 1
    set expr_row $maxrow
    set expr 0
    set fieldvals $stalist
    #set i [llength $fieldvals]

    foreach field $fieldvals {
#puts "the field name is:   $field"

        set Fields($field) 0
        set Sequence($field) ""
        if { [is_expression $field] } {
            set row $expr_row
            incr expr_row
            set col 0
            set expr 1
        } else {
            set row [expr $i % $maxrow + 10]
            set col [expr 2*($i / $maxrow)]
            lappend Possible $field
            }

        radiobutton $f.cb$i \
            -text $field \
            -anchor w \
            -value $field \
            -relief flat \
            -variable Savetype

        set Checkbutton($field) $f.cb$i
        incr i
        }
        incr i -1
        set y 1
        set j 1
        set k 1
        set l 1
        set  maxcol 25
        set up 0
        #set col 0
        set row [expr $up % $maxcol]
        set col [expr $up / $maxcol + 10 ]
        set step [expr $row + 1]
```

```
            for {set y 0} {$y <= $i} {incr y} {
               set row [expr $up % $maxcol]
               set col [expr $up / $maxcol + 10 ]
               blt::table $f $f.cb$y $row,$col -anchor w
               incr up
            }

         radiobutton $f.rbother \
            -text "other" \
            -anchor w \
            -value other \
            -relief flat \
            -variable Savetype \
            -command "EnterNewTime $current; destroy $w"

      label $f.label -text "" -anchor center
      blt::table $f $f.rbother $row,$col -anchor w -fill x
      return $Ok
}

proc which_way {} {
   global num wanted gtime nend nt w

#puts " we are in which_way "
   if {$gtime == 2} {
      set gtime -
      GetFields
   }
   if {$gtime == 4} {
      set gtime -
      #EditValues
      set num 0
      Edits
   }
}

proc time_span {} {
   global Srtdbstg wanted gtime nend nt w Savetype  \
        start end RIdbj5 short_set short_set2

   set time_idx ""
   #puts "save type is $Savetype"
   if {$nt == 1} {
      $w.st configure -state disabled
      $w.et configure -state active
      set start $Savetype
      puts "start is $start"
      foreach station $wanted {
         set DbSta [dbsubset $Srtdbstg "sta == \"$station\""]
         set short_set [ dbsubset $DbSta "time >= \"$start\"" ]
         set rec_cnt [dbquery $short_set dbRECORD_COUNT]
         #puts "rec1 is $rec_cnt"
      }

      set nt -
   }
   if {$nend == 1} {
       $w.st configure -state disabled
       $w.et configure -state disabled
       set end $Savetype
       #puts "end is $end"
        destroy $w
      foreach station $wanted {
          set DbSta [dbsubset $short_set "sta == \"$station\""]
          set short_set2 [ dbsubset $DbSta "time <= \"$end\"" ]
          set rec_cnt [dbquery $short_set2 dbRECORD_COUNT]
          puts "rec1 is $rec_cnt"
       }

       set nend -
       set gtime 4
       GetFields
    }

}

proc EnterNewTime {current} {
    global Savetype f newtype gtime

    $f.rbother flash
    #set newtype ""
    set title "Enter New Time"
    set w [uniqueW ""]
    toplevel $w
    wm title $w $title
    frame $w.w1
    label $w.w1.newtime -text "Enter new $current  (mm/dd/yy): "
    entry $w.w1.entnewtime -width 25 -textvariable newtype($gtime)
    pack $w.w1 -side top
    pack $w.w1.newtime $w.w1.entnewtime -side left
    frame $w.w2
    button $w.w2.cancel \
        -text "cancel" \
        -command "destroy $w"
    pack $w.w2 -side bottom
    pack $w.w2.cancel -side left -fill x

    bind $w.w1.entnewtime <Key-Return> "gosomewhere; destroy $w"

}

proc NewStartTime {current} {
   global Possible SaveType Ok nt nend f w newtype wanted SelectST

   set nt 1
   set nend -
   CheckTimes $current
}

proc NewEndTime {current} {
    global Possible SaveType gtime nend f w newtype wanted SelectET

    set nend 1
    set gtime 2
    CheckTimes $current
}

proc gosomewhere {} {
   global SaveType newtype gtime SelectedSta SelectedST \
                SelectedET editlist

   set SelectedET $newtype($gtime)
   #puts "the new time is : $newtype($gtime)"
   if {$gtime == 1} {
        set SelectedST $newtype(0)
        set editlist "Editing $SelectedSta from $SelectedST to $SelectedET"
        GetFields
        }
    }

    proc setsavelabel { w } {
        global Savetype
        if { $Savetype == 0 } {
            $w config -text "Database Name"
        } elseif { $Savetype == 1 } {
            $w config -text "Text file path"
        } elseif { $Savetype == 2 } {
            $w config -text "Text file path"
        } elseif { $Savetype == 3 } {
            $w config -text "Table name"
        }
    }


    proc table_fields {} {
        global Fields N Sequence Checkbutton request
        global Ok wanted name Possible stalist get_sta

        set title $name
        set w [uniqueW ""]
        toplevel $w
        wm title $w $title

        set maxrow 20
        set maxfields 10
        set lastrow [expr $maxrow+20]
        set span [expr $maxfields / $maxrow + 1]

        frame $w.f1
        button $w.f1.all \
            -text "all" \
            -command "check_all 1"
        button $w.f1.none \
            -text "none" \
            -command "check_all 0"

        pack $w.f1 -side top -fill x
        pack $w.f1.all $w.f1.none -side left -fill x -expand yes

        frame $w.f2
        button $w.f2.cancel \
            -text "cancel" \
            -command "set Ok 0 ; destroy $w"
        button $w.f2.ok \
            -text "ok" \
            -command "set Ok 1; destroy $w"
        pack $w.f2 -side bottom -fill x
        pack $w.f2.cancel $w.f2.ok -side left -fill x -expand yes

        set f $w.f3
        frame $f
        pack $f -side top -fill x

        if { [info exists Fields]} {
            unset Fields
            unset Sequence
            unset Checkbutton
            }

        set i 0
        set N 1
    set expr_row $maxrow
    set expr 0

    set Possible ""
#########################################################
# This is the list of fields in stapars. later, this will
# be changed to be part of the parameter file.
##########################################################

    set fieldvals $stalist
    foreach field $fieldvals {
#puts "the field name is:   $field"
        set Fields($field) 0
        set Sequence($field) ""
        if { [is_expression $field] } {
            set row $expr_row
            incr expr_row
            set col 0
            set expr 1
        } else {
            set row [expr $i % $maxrow + 10]
            set col [expr 2*($i / $maxrow)]
            lappend Possible $field
            }
        label $f.l$i -textvariable Sequence($field) -width 2
        checkbutton $f.cb$i \
            -text $field \
            -anchor w \
            -relief flat \
            -variable Fields($field) \
            -command "fix_order \{$field\}"
        set Checkbutton($field) $f.cb$i
        blt::table $f $f.l$i  $row,$col -anchor e -fill x
        incr col
        if { $expr } {
            blt::table $f $f.cb$i $row,$col -columnspan 25 -anchor w -fill x
        } else {
            blt::table $f $f.cb$i $row,$col -anchor w -fill x
            }
        incr i
        }
#puts "possible is : $Possible"
#puts "the sequence is: $N"

    global Ok
#    check_current $request

    update
    grab set $w
    tkwait window $w
    grab release $w
    global Ok wanted
    set wanted ""
    if { $Ok } {
        set wanted ""
        foreach field [array names Fields] {
            if { $Fields($field) } {
                lappend wanted $field
                }
            }
        set wanted [lsort -command by_value $wanted]
puts "new list in table_field  has: $wanted"
        which_way
        return $wanted
    } else {
        return ""
        }
}

proc CustomDesign { } {
   global sublist net_name pattern network
   set sublist ""
   set Vdbin [dbopen $pattern/$net_name r+]
   set Vdbst [dblookup $Vdbin 0 site 0 0]
   set rec_cnt [dbquery $Vdbst dbRECORD_COUNT]
   set cnt 0
   while {$cnt < $rec_cnt} {
    set sublist "PushMe"
    incr cnt
   }
   GetButtons $sublist
}

proc GetNextSta { } {
   global Idbstc Vdbin stalist sublist TField RIdbj5 get_sta  \
          wanted SelectedSta name net_name pattern network rec_cnt Vdbst
   global list_sta sfile tmpfile vfile clist ffile stafile \
        cout nout fout tout ovout sout

   set name "Station"
   set sublist ""
   set TField "no"
   set Idbin [dbopen $pattern/$net_name r+]
   set Idbaff [dblookup $Idbin 0 affiliation 0 0]
   set Idbst [dblookup $Idbin 0 site 0 0]
   set Idbsc [dblookup $Idbin 0 sitechan 0 0]
   set Idbnet [dblookup $Idbin 0 network 0 0]
   set Idbins [dblookup $Idbin 0 instrument 0 0]
   set Idbsen [dblookup $Idbin 0 sensor 0 0]
   set Idbstg [dblookup $Idbin 0 stage 0 0]
   set Idbcal [dblookup $Idbin 0 calibration 0 0]

   set rec_cnt [dbquery $Idbaff dbTABLE_NAME]
   puts "the record count is: $rec_cnt"
   set Idbstc [dbjoin $Idbcal $Idbstg]

   set RIdbj1 [dbjoin $Idbstc $Idbsen]
   set RIdbj2 [dbjoin $RIdbj1 $Idbins]
   set RIdbj3 [dbjoin $RIdbj2 $Idbst]
   set RIdbj3a [dbjoin $RIdbj3 $Idbsc]
   set RIdbj4 [dbjoin $RIdbj3a $Idbaff]
   set RIdbj5 [dbjoin $RIdbj4 $Idbnet]

              lappend fout $sfile
              lappend cout $clist
              lappend tout $ffile
              lappend ovout $tmpfile
              lappend nout $vfile
              lappend sout $stafile
}

proc GetStation { w } {
   global Idbstc Vdbin stalist sublist TField RIdbj5 get_sta  \
          wanted SelectedSta name net_name pattern network rec_cnt Vdbst
   global list_sta


puts "this is GetStation!"
```

```
    set name "Station"
    set sublist ""
    set TField "no"
    set Idbin [dbopen $pattern/$net_name r+]
    set Idbaff [dblookup $Idbin 0 affiliation 0 0]
    set Idbst [dblookup $Idbin 0 site 0 0]
    set Idbsc [dblookup $Idbin 0 sitechan 0 0]
    set Idbnet [dblookup $Idbin 0 network 0 0]
    set Idbins [dblookup $Idbin 0 instrument 0 0]
    set Idbsen [dblookup $Idbin 0 sensor 0 0]
    set Idbstg [dblookup $Idbin 0 stage 0 0]
    set Idbcal [dblookup $Idbin 0 calibration 0 0]

    set rec_cnt [dbquery $Idbaff dbTABLE_NAME]
    puts "the record count is: $rec_cnt"
    set Idbstc [dbjoin $Idbcal $Idbstg]

    set RIdbj1 [dbjoin $Idbstc $Idbsen]
    set RIdbj2 [dbjoin $RIdbj1 $Idbins]
    set RIdbj3 [dbjoin $RIdbj2 $Idbst]
    set RIdbj3a [dbjoin $RIdbj3 $Idbsc]
    set RIdbj4 [dbjoin $RIdbj3a $Idbaff]
    set RIdbj5 [dbjoin $RIdbj4 $Idbnet]

    set rec_cnt [dbquery $Idbstg dbRECORD_COUNT]
    set cnt 0
    while {$cnt < $rec_cnt} {
    set get_sta [dbgetv $Idbstg 0 $cnt sta]
    set compare [lsearch $sublist $get_sta]
    if {$compare == -1} {
       lappend sublist $get_sta
    } else {
    }
    incr cnt
    }
    set stalist $sublist
    #GetButtons $sublist
    #JoinTheSta
    table_fields
    set SelectedSta $wanted
    set list_sta $wanted
    if {$wanted != ""} {
    #foreach one $wanted {
    #   set Rdbstc [dbsubset $RIdbj5 "sta ==\"$one\""]
    #}
    #set rec_cnt [dbquery $Rdbstc dbRECORD_COUNT]
    #puts "rec_cnt in GetStation  is : $rec_cnt"
    $w.ebs configure -state disabled
    $w.st configure -state active
    }
}

proc JoinTheSta {} {
  global name wanted pattern rec_cnt RIdbj5 net_name network Idbstc stalist
  set name "Station"
  table_fields
  #set StaTimeList $wanted
  foreach one $wanted {
   set Rdbstc [dbsubset $RIdbj5 "sta ==\"$one\""]
  }
   set rec_cnt [dbquery $Rdbstc dbRECORD_COUNT]
   puts "rec_cnt is : $rec_cnt"
  GetTime $pattern
```

```
}

proc JoinTheTime {} {
  global wanted name pattern rec_cnt RIdbj5 stalist
  set name "Time"
  table_fields
  set EditTimeList $wanted
  foreach that $wanted {
     set Rdbstc [dbsubset $RIdbj5 "time ==\"$that\""]
  }
  set rec_cnt [dbquery $Rdbstc dbRECORD_COUNT]
  puts "rec_cnt is ::: $rec_cnt"
  GetFields
}

proc EditDbWindow { } {
   global w gtime rec_cnt RIdbj5 net_path dfile net_name network pattern

   puts "this is EditDbWindow!"

   set gtime -
    set w [uniqueW ""]
    toplevel $w
    set name "Edit Database Window"
    wm title $w $name
    wm geometry $w +200+300
#     set mb [menu_setup $w]
   set Idbin [dbopen $pattern/$net_name r+]
   set Idbaff [dblookup $Idbin 0 affiliation 0 0]
   set Idbst [dblookup $Idbin 0 site 0 0]
   set Idbsc [dblookup $Idbin 0 sitechan 0 0]
   set Idbnet [dblookup $Idbin 0 network 0 0]
   set Idbins [dblookup $Idbin 0 instrument 0 0]
   set Idbsen [dblookup $Idbin 0 sensor 0 0]
   set Idbstg [dblookup $Idbin 0 stage 0 0]
   set Idbcal [dblookup $Idbin 0 calibration 0 0]

   set rec_cnt [dbquery $Idbaff dbTABLE_NAME]
   puts "the record count is: $rec_cnt"
   set Idbstc [dbjoin $Idbcal $Idbstg]

   set RIdbj1 [dbjoin $Idbstc $Idbsen]
   set RIdbj2 [dbjoin $RIdbj1 $Idbins]
   set RIdbj3 [dbjoin $RIdbj2 $Idbst]
   set RIdbj3a [dbjoin $RIdbj3 $Idbsc]
   set RIdbj4 [dbjoin $RIdbj3a $Idbaff]
   set RIdbj5 [dbjoin $RIdbj4 $Idbnet]

    button $w.ebs -text "EditByStation" -relief ridge \
                                        -command "GetStation $w"
    button $w.st -text "Start Time" -relief ridge \
                                    -state disabled \
                                    -command "set gtime 0; GetTime time"
    button $w.et -text "End Time" -relief ridge \
                                    -state disabled \
                                    -command "set gtime 1; GetTime endtime"
    button $w.dismiss    -text Dismiss   -command "destroy $w"

        #$mb            2,0 -columnspan 100     -anchor w -fill x

    blt::table $w \
        $w.ebs          3,2                     -anchor w -fill x \
        $w.st           3,4                     -anchor w -fill x \
        $w.et           3,6                     -anchor w -fill x \
        $w.dismiss      30,0 -columnspan 100    -fill x
}

proc GetFields {} {
   global Srtdbstg gtime pattern sublist Pf TField tmlist RIdbj5 output\
         name rec_cnt network net_name Idbstc stalist Possible
   global short_set short_set2
puts "this is GetFields!"

   set field_names [ dbquery $RIdbj5 dbTABLE_FIELDS ]
   set DbFields [pfgetlist @$Pf#TableFields]
   set sublist ""
   set tmlist ""
   set TField "no"
   pfput %${Pf}#band E $DbFields
   set field_names [ dbquery $short_set2 dbTABLE_FIELDS ]
   puts $field_names
   set rec_cnt [dbquery $Idbstc dbRECORD_COUNT]
   set cnt 0
   set Possible $DbFields
   set stalist $DbFields
   set name "Field Names"
   set gtime 4
   table_fields
}

proc GetTime {now} {
   global w Ok Idbaff Idbstc wanted name tmlist pattern stalist sublist \
          Srtdbstg gtime TField RIdbj5 output rec_cnt network net_name

puts "this is GetTime"

   set name "Time"
   set sublist ""
   set tmlist ""
   set TField "no"
   puts "the record count is: $rec_cnt"
  puts "now is $now"

   set Srtdbstg [dbsort $RIdbj5 $now]
   set field_names [ dbquery $Srtdbstg dbTABLE_FIELDS ]

   set rec_cnt [dbquery $Idbstc dbRECORD_COUNT]
   set cnt 0
   loop cnt 0 $rec_cnt {
   #set efile [dbgetv $Srtdbstg 0 $cnt stage.$now]
   set dfile [dbgetv $Srtdbstg 0 $cnt $now]
   #set dfile [dbgetv $Idbstc 0 $cnt sta]
   set NewTime [ strtime $dfile ]
   set compare [lsearch $sublist $dfile]
   #set NewTime "$dfile $NewTime"
    #set ep [epoch "$yrday$jday"]
    #set year [yearday "$ep"]
    #set date [strtime "$ep"]
    #set stdate [strdate "$ep"]

      if {$compare == -1} {
         lappend sublist $dfile
         lappend tmlist $NewTime
      }
   }
   set stalist $tmlist

   if {$gtime == 0} {
          NewStartTime $now
     } else {
          NewEndTime $now
     }

}

proc GetTimeButtons { sublist } {
        global stalist pattern Possible new clock newlist search
        set g [uniqueW ""]
        set Possible $sublist
        set stalist $sublist
        toplevel $g
        set name $pattern
        wm title $g $name
        wm geometry $g +200+300
        label $g.space
        frame $g.bar -relief raised -bd 2
        frame $g.dummy
        pack $g.bar $g.dummy -side top -fill x
        menubutton $g.bar.view -text View -menu $g.bar.view.menu
        menubutton $g.bar.help -text Help -menu $g.bar.help.menu
        pack $g.bar.view -side left
        pack $g.bar.help -side right
        menu $g.bar.view.menu
        menu $g.bar.help.menu
        $g.bar.view.menu add command -label "Arrange" -underline 0 -command "JoinTheTime;
destroy $g"
        $g.bar.help.menu add command -label "under construction" -command "uc"
        button $g.dismiss -text Dismiss -command "destroy $g"
        pack $g.dismiss -side bottom -fill x
        tk_menuBar $g.bar $g.bar.view $g.bar.help
        focus $g.bar
        #bell

        set i 1
        foreach search $sublist {

                button $g.run($i) \
                        -text $search \
                        -anchor s \
                        -relief ridge \
                        -command "GetFields; destroy $g"


                #pack $s.run($i) -side left -fill x
                #$s.text insert end $new
                incr i
        }
        incr i -1
        set j 1
        set k 1
        set l 1
        set  maxcol 8
        set up 0
        set col [expr $up % $maxcol]
        set row [expr $up / $maxcol + 10 ]
        set step [expr $row + 1]

             for {set y 1} {$y <= $i} {incr y} {
                set col [expr $up % $maxcol]
                set row [expr $up / $maxcol + 10 ]
                blt::table $g.dummy $g.run($y) $row,$col -anchor center -fill x
                incr up
                }
        focus $g.bar
}

proc GetButtons { sublist } {
        global  pattern Possible stalist new clock newlist search
        set g [uniqueW ""]
        toplevel $g
        set Possible $sublist
        set stalist $sublist
        set name $pattern
        wm title $g $name
        wm geometry $g +200+300
        label $g.space
        frame $g.bar -relief raised -bd 2
        frame $g.dummy
        pack $g.bar $g.dummy -side top -fill x
        menubutton $g.bar.view -text View -menu $g.bar.view.menu
        menubutton $g.bar.help -text Help -menu $g.bar.help.menu
        pack $g.bar.view -side left
        pack $g.bar.help -side right
        menu $g.bar.view.menu
        menu $g.bar.help.menu
        $g.bar.view.menu add command -label "Arrange" -underline 0 -command "JoinTheSta; d
estroy $g"
        $g.bar.help.menu add command -label "under construction" -command "uc"
        button $g.dismiss -text Dismiss -command "destroy $g"
        pack $g.dismiss -side bottom -fill x
        tk_menuBar $g.bar $g.bar.view $g.bar.help
        focus $g.bar
        #bell

        set i 1
        foreach search $sublist {

                button $g.run($i) \
                        -text $search \
                        -anchor s \
                        -relief ridge \
                        -command "EditDb $search"


                #pack $s.run($i) -side left -fill x
                #$s.text insert end $new
                incr i
        }
        incr i -1
        set j 1
        set k 1
        set l 1
        set  maxcol 8
        set up 0
        set col [expr $up % $maxcol]
        set row [expr $up / $maxcol + 10 ]
        set step [expr $row + 1]

             for {set y 1} {$y <= $i} {incr y} {
                set col [expr $up % $maxcol]
                set row [expr $up / $maxcol + 10 ]
                blt::table $g.dummy $g.run($y) $row,$col -anchor center -fill x
                incr up
             }
        focus $g.bar
}
```

```
proc EditValues {} {
   global pattern net_name short_set2 short_set wanted cnt Srtdbstg RIdbj5 \
        pattern1 ck_cnt sfile ffile clist vfile Dbcnt

   set blist ""
   set sfile ""
   set clist ""
   set ffile ""
   set vfile ""
   set max $rec_cnt
   set max [expr ($max * 10)]

   set ncnt 2

    set tnt 3
   set item [lindex $wanted 0]

   puts "wanted in EditV is $wanted"

   foreach step $wanted {
   loop cnt 0 $rec_cnt {
   set dfile [dbgetv $short_set2 0 $cnt $step]
   set tfile [dbgetv $short_set2 0 $cnt time]
   set cfile [dbgetv $short_set2 0 $cnt chan]
   set tfile [ strtime $tfile ]
      set n [uniqueN]
        incr tnt
#puts "file is $tfile"
   set string_cat "$dfile$cfile$tfile"
   set compare [lsearch $blist $string_cat]
      if {$compare == -1} {
         lappend blist $string_cat

        lappend sfile $step
        lappend clist $cfile
        lappend ffile $tfile
        lappend vfile $dfile
      }
    incr cnt
    }
   }

 EditWindow
 }

 proc EditWindow {} {
   global Entry clist name f lblist ffile vfile net_name pattern1 sfile
   global num length entry SelectedSta Selection blist
   global fields rec_cnt stafile station selection
   global Dbsite Dbchan Dbvalue Dbfield Dbtime

   set f [uniqueW ""]
   set n [uniqueN]

   set lblist ""
   set i 0
   set station [lindex $SelectedSta $i]
   #toplevel $f

        toplevel $f
        set name "Editing $pattern1/$net_name"
        wm title $f $name
        label $f.space
```

```
     frame $f.bar
     frame $f.dummy
     pack $f.bar -side top -fill x
     pack $f.dummy -side bottom -fill x

     menubutton $f.bar.file -text File -menu $f.bar.file.menu
     menubutton $f.bar.help -text Help -menu $f.bar.help.menu
     pack $f.bar.file -side left
     pack $f.bar.help -side right
     menu $f.bar.file.menu
     $f.bar.file.menu add command -label "Exit" -command "uc"
     menu $f.bar.help.menu
     $f.bar.help.menu add command -label "under construction" -command "uc"
button $f.can -text "Cancel" -command "destroy $f"

button $f.nex -text "Next Station" -state disabled \
                    -command "GetNextSta; Edits"
button $f.execute -text Execute -state disabled \
                    -command "CheckVals"
pack $f.can $f.nex $f.execute -side left -in $f.dummy -fill x

set xframe [frame $f.ok -relief groove -bd 2]
pack $xframe -side top

frame $f.main -relief raised
pack $f.main -side bottom


menubutton $f.mfile \
     -text "File" \
     -menu $f.mfile.menu

menu $f.mfile.menu

$f.mfile.menu add command \
     -label "Exit" \
     -command {destroy $f; cleanup}

menubutton $f.mhelp \
     -text "Help" \
     -menu $f.mhelp.menu

menu $f.mhelp.menu

$f.mhelp.menu add command \
     -label "editing procedure" \
     -command "uc"

label $f.title \
     -text "Editing $pattern1/$net_name" \
     -foreground blue

label $f.sta \
     -text "Station: $station" \
     -foreground blue

label $f.times \
     -relief raised \
     -text "Time"

lappend lblist $f.list_time

listbox $f.list_time \
     -relief ridge \
          -height 10 \
          -selectmode single \
          -yscrollcommand "$f.scrb set"

     label $f.chans \
          -relief raised \
          -text "Chan"

     lappend lblist $f.chan_list

     listbox $f.chan_list \
          -relief ridge \
          -height 10 \
          -selectmode single \
          -yscrollcommand "$f.scrb set"

     label $f.fields \
          -relief raised \
          -text "Field"

     lappend lblist $f.field_list

     listbox $f.field_list \
          -relief ridge \
          -height 10 \
          -selectmode single \
          -yscrollcommand "$f.scrb set"

     label $f.values \
          -relief raised \
          -text "Field Value"

     lappend lblist $f.value_list

     set name $f.value_list$n
     listbox $f.value_list \
          -relief ridge \
          -height 10 \
          -selectmode single \
          -yscrollcommand "$f.scrb set"

      bind $f.value_list  <Any-ButtonRelease> "set_entry %W $name"

     scrollbar $f.scrb \
          -width 10 \
          -command {multi_scroll \
               $lblist}

      bind $f.scrb <Any-Button> "+set Selection($f) {}"

     label $f.nothing \
          -text "                     "


     label $f.news \
          -text "New Field Value:"

     entry $f.entrys \
          -textvariable Entry($f)

     bind $f.entrys <Return> "save_edit; EditDb; CautionWin"
     label $f.message \
          -text "Message:"
```

```
   entry $f.enter \
        -relief ridge \
        -state disabled \
        -textvariable entry

   blt::table $xframe \
        $f.title        2,1 -anchor w \
        $f.sta          3,1 -anchor w  \
        $f.times        4,2 -anchor w -fill x \
        $f.chans        4,3 -anchor w -fill x \
        $f.fields       4,4 -anchor w -fill x \
        $f.values       4,5 -anchor w -fill x \
        $f.list_time    5,2 -anchor w -fill x \
        $f.chan_list    5,3 -anchor w -fill x \
        $f.field_list   5,4 -anchor w -fill x \
        $f.value_list   5,5 -anchor w -fill x \
        $f.scrb         5,6 -fill y \
        $f.news         7,1 -anchor w \
        $f.entrys       8,1 -anchor w \
        $f.message      9,1 -anchor w \
        $f.enter        10,1 -columnspan 100 -anchor w -fill x

#puts "lblist is : $lblist"
#puts "clist is:  $clist"
   incr rec_cnt -1
   set t 0
   set list $clist
   foreach ten $sfile {
   set cnt [lindex $list $t]
   set lmax [llength $list]
#   puts "lmax is :  $lmax"


         catch {$f.field_list insert end $Dbfield($ten,$cnt)}
            catch {$f.chan_list insert end $Dbchan($ten,$cnt)}
            catch {$f.list_time insert end $Dbtime($ten,$cnt)}
            catch {$f.value_list insert end $Dbvalue($ten,$cnt)}
        set list [lreplace $list 0 0]

   }



    global Column_label Column Parent Expression Name
    set Parent($name) $f
   if {[llength $SelectedSta] > 1} {
        set SelectedSta [lreplace $SelectedSta 0 0]
        $f.nex configure -state normal
        $f.enter configure -state normal
        $f.enter configure -background red
        $f.enter configure -foreground white
        $f.enter configure -font -*-bookman-*-*-*-*-14-*-*-*-*-*-*-*
        set entry "Stations: $SelectedSta left to be edited. \
             Use Next Station Button to get next station."
   }
#   puts "num is $num"
#   puts "length is $length"

   if {$num == $length} {
      $f.nex configure -state disabled
      $f.execute configure -state normal
        $f.enter configure -background red
        $f.enter configure -foreground white
        $f.enter configure -font -*-bookman-*-*-*-*-14-*-*-*-*-*-*-*
        set entry "No more stations to edit. Use Execute \
                  to implement your changes."
     }
        incr num

 #CheckVals

 }

 proc CheckVals {} {
    global f lblist sfile ffile vfile SelectedSta net_name pattern1 clist
    global tmpfile stafile
    global cout fout tout ovout nout sout
    global Dbsite Dbchan Dbvalue Dbfield Dbtime

    set f [uniqueW ""]
    set n [uniqueN]

    set lblist ""
    set i 0
    set field_list "sta chan time field oldvalue \
                newvalue"

    set station [lindex $SelectedSta $i]
         toplevel $f
         set name "Editing $pattern1/$net_name"
         wm title $f $name
         label $f.space
         frame $f.bar
         frame $f.dummy
         pack $f.bar -side top -fill x
         pack $f.dummy -side bottom -fill x

         menubutton $f.bar.file -text File -menu $f.bar.file.menu
         menubutton $f.bar.help -text Help -menu $f.bar.help.menu
         pack $f.bar.file -side left
         pack $f.bar.help -side right
         menu $f.bar.file.menu
         $f.bar.file.menu add command -label "Exit" -command "uc"
         menu $f.bar.help.menu
         $f.bar.help.menu add command -label "under construction" -command "uc"
    button $f.can -text "Cancel" -command "destroy $f"
    button $f.nex -text "Next Station" -state disabled \
                        -command "GetNextSta; Edits"
    button $f.execute -text Execute -state disabled

    pack $f.can $f.nex $f.execute -side left -in $f.dummy -fill x

    set xframe [frame $f.ok -relief groove -bd 2]
    pack $xframe -side top

    frame $f.main -relief raised
    pack $f.main -side bottom

    menubutton $f.mfile \
         -text "File" \
         -menu $f.mfile.menu

    menu $f.mfile.menu

    $f.mfile.menu add command \
         -label "Exit" \
         -command {destroy $f; cleanup}
```

```
menubutton $f.mhelp \
     -text "Help" \
     -menu $f.mhelp.menu

menu $f.mhelp.menu

$f.mhelp.menu add command \
     -label "editing procedure" \
     -command "uc"

label $f.title \
     -text "Editing $pattern1/$net_name" \
     -foreground blue

label $f.sta \
     -text "Station: $station" \
     -foreground blue

set row 2
set col 1
foreach field $field_list {

   set prow [expr ($row + 1)]
   set pcol $col
   set max [llength $field]
   set max [expr ($max + 10)]

   label $f.ck$field \
     -relief raised \
     -text $field

   lappend lblist $f.lb$field

   listbox $f.lb$field \
     -relief ridge \
     -width $max \
     -height 10 \
     -selectmode single \
     -yscrollcommand "$f.sb set"

   blt::table $xframe \
     $f.ck$field     $row,$col        -anchor w -fill x \
     $f.lb$field     $prow,$pcol      -anchor w -fill x

set t 0
set list $clist
foreach ten $sfile {
set cnt [lindex $list $t]
set lmax [llength $list]
puts "lmax is :  $lmax"

      catch {$f.field_list insert end $Dbfield($ten,$cnt)}
         catch {$f.chan_list insert end $Dbchan($ten,$cnt)}
         catch {$f.list_time insert end $Dbtime($ten,$cnt)}
         catch {$f.value_list insert end $Dbvalue($ten,$cnt)}
     set list [lreplace $list 0 0]

}

#       switch $field {
#
#        sta     { foreach four $sout {
#                      $f.lb$field insert end $four
#                   }}
#        chan    { foreach four $cout {
#                     $f.lb$field insert end $four
#                   }}
#       time     { foreach four $tout {
#                     $f.lb$field insert end $four
#                   }}
#       field    { foreach four $fout {
#                     $f.lb$field insert end $four
#                   }}
#       oldvalue    { foreach four $ovout {
#                     $f.lb$field insert end $four
#                   }}
#       newvalue    { foreach four $nout {
#                     $f.lb$field insert end $four
#                   }}
#      }


      #incr row
      incr col

   }
   incr pcol
   scrollbar $f.sb \
        -width 10 \
        -command {multi_scroll $lblist}

   blt::table $xframe \
        $f.sb           $prow,$pcol     -anchor w -fill y

}

##############################################
# grabs the selected fields from the database
# also gets corresponding channel and time.
##############################################

proc Edits {} {
   global RIdbj5 Srtdbstg DbSta short_set short_set2 start end
   global wanted blist list_sta pattern1 sfile ffile clist vfile
   global Dbcnt tmpfile num length station
   global fields rec_cnt pattern net_name EndTime stafile
   global Dbsite Dbchan Dbvalue Dbfield Dbtime

   set blist ""
   set cfile ""
   set dfile ""
   set tfile ""
   set sfile ""
   set clist ""
   set ffile ""
   set vfile ""
   set stafile ""

   set length [llength $list_sta]
   set length [expr ($length -1)]
   puts "list_sta is $list_sta"

   set station [lindex $list_sta $num]

      puts "start is $start"
      puts "end is $end"
      puts "station is $station"
```

```
    set Idbin [dbopen $pattern/$net_name r+]
    set Idbaff [dblookup $Idbin 0 affiliation 0 0]
    set Idbst [dblookup $Idbin 0 site 0 0]
    set Idbsc [dblookup $Idbin 0 sitechan 0 0]
    set Idbnet [dblookup $Idbin 0 network 0 0]
    set Idbins [dblookup $Idbin 0 instrument 0 0]
    set Idbsen [dblookup $Idbin 0 sensor 0 0]
    set Idbstg [dblookup $Idbin 0 stage 0 0]
    set Idbcal [dblookup $Idbin 0 calibration 0 0]

    set Idbstc [dbjoin $Idbcal $Idbstg]

    set RIdbj1 [dbjoin $Idbstc $Idbsen]
    set RIdbj2 [dbjoin $RIdbj1 $Idbins]
    set RIdbj3 [dbjoin $RIdbj2 $Idbst]
    set RIdbj3a [dbjoin $RIdbj3 $Idbsc]
    set RIdbj4 [dbjoin $RIdbj3a $Idbaff]
    set RIdbj5 [dbjoin $RIdbj4 $Idbnet]

    set rec_cnt [dbquery $RIdbj5 dbRECORD_COUNT]
#    puts "the record count is: $rec_cnt"
    set Srtdbstg [dbsort $RIdbj5 sta]
       set Jdbsta [dbsubset $Srtdbstg "sta == \"$station\""]

       if {[catch {set StartTime [dbsubset $Jdbsta "time >= \"$start\""]} \
             st_err]} {
         puts "$station does not have data for this time."
       }
       if {[catch {set EndTime [dbsubset $StartTime "time <= \"$end\""]} \
             et_err]} {
         puts "$station does not have data for this time."
       }

       set rec_cnt [dbquery $EndTime dbRECORD_COUNT]
#       puts "the endtime $EndTime"
       set fields $wanted
     foreach step $wanted {
       set cnt 0
       loop cnt 0 $rec_cnt {
       #set rec_cnt [dbquery $EndTime dbRECORD_COUNT]
#       puts "the rec is $rec_cnt"
          if {[catch {set dfile [dbgetv $EndTime 0 $cnt $step]} \
             df_err]} {
             puts "$station does not have data for this time $end."
          }
          if {[catch {set tfile [dbgetv $EndTime 0 $cnt time]} \
             tf_err]} {
             puts "$station does not have data for $end."
          } else {
             set tfile [ strtime $tfile ]
          }
          if {[catch {set cfile [dbgetv $EndTime 0 $cnt chan]} \
             cf_err]} {
             puts "$station does not have data for this time."
          }
          if {[catch {set site [dbgetv $EndTime 0 $cnt sta]} \
             cf_err]} {
             puts "$station does not have data for this time."
          }
          set n [uniqueN]
#set Dbcnt($step,$cnt) $dfile
#puts "db count($step,$cnt) is $Dbcnt($step,$cnt)"
```

```
          set string_cat $dfile$cfile$tfile

#puts "string_cat is $string_cat"

          set compare [lsearch $blist $string_cat]

               lappend blist $string_cat

             if {($compare == -1) && $dfile >= 0} {
   set Dbcnt($step,$cnt) $dfile

#puts "db count($step,$cnt) is $Dbcnt($step,$cnt)"

   set Dbfield($step,$cnt) $step
   set Dbvalue($step,$cnt) $dfile
   set Dbchan($step,$cnt) $cfile
   set Dbtime($step,$cnt) $tfile
   set Dbsite($step,$cnt) $site

#puts "dbfield is [array names Dbfield]"

               lappend sfile $step
               lappend clist $cnt
               lappend ffile $tfile
               lappend vfile $dfile
               lappend stafile $site
               set tmpfile $vfile
             }

          #incr cnt
      }
    }
   puts "stafile is $stafile"

   EditWindow
}

proc EditDb {} {
   global pattern net_name short_set2 short_set Srtdbstg RIdbj5
   global tmpfile sfile clist ffile vfile Entry f
   global num length entry SelectedSta Selection blist
   global Value station selection EndTime
   global Dbvalue cnt Jdbsta

   set cnt 0
   set Idbin [dbopen $pattern/$net_name r+]
   set Idbaff [dblookup $Idbin 0 affiliation 0 0]
   set Idbst [dblookup $Idbin 0 site 0 0]
   set Idbsc [dblookup $Idbin 0 sitechan 0 0]
   set Idbnet [dblookup $Idbin 0 network 0 0]
   set Idbins [dblookup $Idbin 0 instrument 0 0]
   set Idbsen [dblookup $Idbin 0 sensor 0 0]
   set Idbstg [dblookup $Idbin 0 stage 0 0]
   set Idbcal [dblookup $Idbin 0 calibration 0 0]

   set Idbstc [dbjoin $Idbcal $Idbstg]

   set RIdbj1 [dbjoin $Idbstc $Idbsen]
   set RIdbj2 [dbjoin $RIdbj1 $Idbins]
   set RIdbj3 [dbjoin $RIdbj2 $Idbst]
   set RIdbj3a [dbjoin $RIdbj3 $Idbsc]
   set RIdbj4 [dbjoin $RIdbj3a $Idbaff]
   set RIdbj5 [dbjoin $RIdbj4 $Idbnet]
   set Srtdbstg [dbsort $RIdbj5 sta]
   set Jdbsta [dbsubset $Srtdbstg "sta == \"$station\""]

      set rec_cnt [dbquery $Jdbsta dbRECORD_COUNT]
      set rec_cnt [expr ($rec_cnt - 1)]
      puts "rec_cnt is $rec_cnt"
   set tnt $selection

#   if {$cnt > $rec_cnt} {
#      set cnt [expr ($cnt - $rec_cnt)]
#   }

puts "clist is $clist"

set cnt [lindex $clist $selection]

puts "the cnt is $cnt"
#puts "the selection is $cnt"

   set test $Entry($f)
   set Value [lindex $sfile $selection]
   set tmpfile $vfile
   set Dbvalue($Value,$cnt) $Entry($f)
   set vfile [lreplace $vfile $tnt $tnt $Entry($f)]
   #dbputv $Jdbsta 0 $cnt $Value $Entry($f)

   OtherFields
}

proc OtherFields {} {
   global Dbsite Dbchan Dbvalue Dbfield Dbtime
   global data net_name Entry Value list f cnt EndTime Jdbsta Pf

   pfgetarr fields %${Pf}#field_map
   for_array_keys changed fields {
      if {$Value == $changed} {
          set select $fields($changed)
          foreach one $select {
             lappend list $one
          }
      }
   }

         if {[catch {set string [dbgetv $EndTime 0 $cnt dfile]} \
            cf_err4]} {
            puts "$station does not have data for this time."
         } else {
               set this1 [string length $string]
               set this [expr ($this1 + 1)]
                  set data [dbgetv $EndTime 0 $cnt insname]
               set string [string trimright [string range $data $this end]]
         }
         if {[catch {set rate [dbgetv $EndTime 0 $cnt samprate]} \
            cf_err5]} {
            puts "$station does not have data for this time."
         }


        CalcInstrTbl
        PutBack

    global string2 rfile coil inid name type insdfile

      puts "this is:   $this"
```

```
       puts "string is : $string"

        set name "$rfile:$string"
        set insdfile $rfile$string2
        set dfile $rfile


    pfgetarr logger %${Pf}#LoggersList
       if {$string == "GEOS"} {
          pfgetarr logger2 %${Pf}#LoggersList#$string
          for_array_keys data logger2 {
             set cv $logger2($data)
puts "what's cv1:  $cv"
             set xcalib [expr ((1.0/$cv)/$coil)]
             set cal [expr ($xcalib * 10e06)]
puts "coil is : $coil and cal is : $cal"
puts "what's xcalib1: $xcalib"
             dbputv $Jdbsta 0 $cnt calib $cal
          }
       } else {
            for_array_keys Data logger {
              if {$Data == $string} {
                 set cv $logger($Data)
puts "what's cv:  $cv"
set tcl_precision 17

                 set xcalib [expr (1.0/$cv)]
                 set xcalib [expr ($xcalib/$coil)]
puts "what's xcalib2: $xcalib"

                 set cal [expr ($xcalib * 10e06)]
puts "coil2 is : $coil and cal is : $cal"
                 dbputv $Jdbsta 0 $cnt calib $cal
               }
            }
       }
#puts "you die here!"
   #dbputv $Jdbsta 0 $cnt $Value $Entry($f)

puts "coil3 is : $coil and cal is : $cal"
          if {[catch {set stg_gtype [dbgetv $EndTime 0 $cnt gtype]} \
             cf_err5]} {
             puts "$station does not have data for this time."
          } else {
              switch $stg_gtype {
puts "you die here!"

                 sensor  { dbputv $Jdbsta 0 $cnt gnom $coil }
                 digitizer  { dbputv $Jdbsta 0 $cnt gnom $cv }
                 FIR_decimator   { dbputv $Jdbsta 0 $cnt gnom 1.0 }
             }
          }
}

proc CalcInstrTbl {} {
   global this string string2 rfile Value EndTime cnt
   global Pf pattern net_name coil Entry f type inid
   global insdfile name Jdbsta

          if {[catch {set data [dbgetv $EndTime 0 $cnt insname]} \
             cf_err1]} {
             puts "$station does not have data for this time."
          }
          if {[catch {set band [dbgetv $EndTime 0 $cnt band]} \
           cf_err1]} {
           puts "$station does not have data for this time."
        }
        if {[catch {set inid [dbgetv $EndTime 0 $cnt inid]} \
           cf_err1]} {
           puts "$station does not have data for this time."
        }
        if {[catch {set digital [dbgetv $EndTime 0 $cnt digital]} \
           cf_err1]} {
           puts "$station does not have data for this time."
        }
        if {[catch {set rsptype [dbgetv $EndTime 0 $cnt rsptype]} \
           cf_err1]} {
           puts "$station does not have data for this time."
        }
        if {[catch {set dir [dbgetv $EndTime 0 $cnt instrument.dir]} \
           cf_err1]} {
           puts "$station does not have data for this time."
        }
        if {[catch {set ncalib [dbgetv $EndTime 0 $cnt ncalib]} \
           cf_err1]} {
           puts "$station does not have data for this time."
        }
        if {[catch {set ncalper [dbgetv $EndTime 0 $cnt ncalper]} \
           cf_err1]} {
           puts "$station does not have data for this time."
        }
        if {[catch {set type [dbgetv $EndTime 0 $cnt instype]} \
           cf_err2]} {
           puts "$station does not have data for this time."
        }
        if {[catch {set rate2 [dbgetv $EndTime 0 $cnt instrument.samprate]} \
           cf_err3]} {
           puts "$station does not have data for this time."
        }
        if {[catch {set string [dbgetv $EndTime 0 $cnt dfile]} \
           cf_err4]} {
           puts "$station does not have data for this time."
        } else {
                set this1 [string length $string]
                set this [expr ($this1 + 1)]
                   set data [dbgetv $EndTime 0 $cnt insname]
                set string [string trimright [string range $data $this end]]
        }
        if {[catch {set rate [dbgetv $EndTime 0 $cnt samprate]} \
           cf_err5]} {
           puts "$station does not have data for this time."
        }
        if {[catch {set string2 [dbgetv $EndTime 0 $cnt instrument.dfile]} \
           cf_err6]} {
           puts "$station does not have data for this time."
        } else {
                   set string3 [string trimright \
                        [string range $string2 $this1 end]]
                   puts "string2 is:  $string3"
        }
        if {[catch {set chan [dbgetv $EndTime 0 $cnt sensor.chan]} \
           cf_err5]} {
           puts "$station does not have data for this time."
        }

  set inid [expr ($inid + 1)]
  set instr_fid [open $pattern/$net_name.instrument a+]
  set dbout [dbopen $pattern/$net_name r+]
```

```
    set Idbins [dblookup $dbout 0 instrument 0 0]
    set Idbrec [dbquery $Idbins dbRECORD_COUNT]
    set Idbrec [expr ($Idbrec + 1)]
    puts "Idbrec is :$Idbrec"
    set inid $Idbrec


    switch $Value {

 puts "what's val: $Value"

        samprate    { #set rate [dbgetv $EndTime 0 $cnt samprate]
                      dbputv $EndTime 0 $cnt samprate $Entry($f)
    if {[catch {pfgetarr Data %${Pf}#$string} err] != 1} {
        pfgetarr list %${Pf}#$string#Samp
        for_array_keys fir list {
           if { $fir == $Entry($f) } {
              set Fir $list($fir)
          puts "you got it kid!"
           }
        }
    }
                    }
        instrument.samprate   { set rate2 [dbgetv $EndTime \
                            0 $cnt instrument.samprate]
                    }
        instype     { #set type [dbgetv $EndTime 0 $cnt instype]
                      set type $Entry($f)
                    }
    }


 puts "what is type:   $type"

    if {[catch {pfgetarr Sensor %${Pf}#$type} err] != 1} {

        for_array_keys sensor Sensor {

 puts "sensors are: $sensor"

            switch $sensor {
               response   {set rfile $Sensor($sensor)}
               freq       {set freq $Sensor($sensor)}
               damp        {set damp $Sensor($sensor)}
               const       {set coil $Sensor($sensor)}
            }
        }

    set ch_code ""
         ################################################
         # calculate the band and channel codes
         ################################################
         set ck_freq [expr (1 / $freq)]
         set ck_samp $rate2


         if {($ck_samp >= 80.0) && ($ck_freq \
                 < 10.0)} {
             set band "e"
             set ch_code E
         } elseif {($ck_samp >= 10.0) && ($ck_samp < 80.0) \
                 && ($ck_freq < 10.0)} {
             set band "s"
             set ch_code S
        } elseif {($ck_samp >= 80.0) && ($ck_freq \
                >= 10.0)} {
            set band "h"
            set ch_code H
        } elseif {($ck_samp >= 10.0) && ($ck_samp < 80.0) \
                && ($ck_freq >= 10.0)} {
            set band "b"
            set ch_code B
        } elseif {($ck_samp > 1.0) && ($ck_samp < 10.0)} {
            set band "m"
            set ch_code M
        } else {
                puts "no band code for the samprate and corner!"
        }

                    set band_str [string trimright \
                        [string range $chan 1 end]]
        puts "band_str is $band_str"
        append ch_code $band_str
        puts "ch_code is $ch_code"
        dbputv $Jdbsta 0 $cnt dfile $rfile
        #dbputv $Jdbsta 0 $cnt sensor.chan $ch_code
        #dbputv $Jdbsta 0 $cnt chan $ch_code
        #dbputv $Jdbsta 0 $cnt stage.chan $ch_code
        #dbputv $Jdbsta 0 $cnt sitechan.chan $ch_code

    }

        set under "_"
        set name "$rfile:$string"
        set name2 "$rfile$under$string.2"
puts "name2 is $name2"
        set insdfile $rfile$string3
        #set dfile $rfile

    dbputv $Jdbsta 0 $cnt inid $Idbrec

    set count 1
    loop cnt 0 $Idbrec {
        set ck1 [dbgetv $EndTime 0 $cnt insname]
        set ck2 [dbgetv $EndTime 0 $cnt instrument.samprate]
        if {$ck1 != $name} {
           set ck_ans "yes"
        } else {
           set ck_ans "no"
           if {$ck2 != $rate2} {
              incr count
              set ck_ans "yes"
              set insdfile "$rfile$under$string.$count"
              puts "insdfile is $insdfile"
           }
        }

    }
    if {$ck_ans == "yes"} {
       if {[catch {dbaddv $dbout instrument inid $inid \
              insname $name instype $type \
              band $band digital $digital \
              samprate $rate2 ncalib $ncalib \
              ncalper $ncalper dir $dir \
              dfile $insdfile rsptype $rsptype} instr_add_err]} {
              puts "instrument out:      $instr_add_err"
       }
    }
```

```
   ###############################################################
   # now write the new instrument table record
   ###############################################################

   puts "how many mc's"

}

proc RestOfThem {} {
   global Dbsite Dbchan Dbvalue Dbfield Dbtime
   global Entry Value list f cnt EndTime Jdbsta Pf

   switch $Values {
      dir     { set dr [dbgetv $EndTime 0 $cnt $Values]
                     dbputv $Jdbsta 0 $cnt dir $Entry($f)
                     dbputv $Jdbsta 0 $cnt instrument.dir $Entry($f)
                }
      instrument.dir  { set dr [dbgetv $EndTime 0 $cnt $Values]
                     dbputv $Jdbsta 0 $cnt dir $Entry($f)
                     dbputv $Jdbsta 0 $cnt instrument.dir $Entry($f)
                }
      jdate     { set jd [dbgetv $EndTime 0 $cnt $Values]
                     dbputv $Jdbsta 0 $cnt jdate $Entry($f)
                }
      ondate     { set on [dbgetv $EndTime 0 $cnt $Values]
                     dbputv $Jdbsta 0 $cnt ondate $Entry($f)
                     dbputv $Jdbsta 0 $cnt sitechan.ondate $Entry($f)
                }
      sitechan.ondate { set on [dbgetv $EndTime 0 $cnt $Values]
                     dbputv $Jdbsta 0 $cnt ondate $Entry($f)
                     dbputv $Jdbsta 0 $cnt sitechan.ondate $Entry($f)
                }
      offdate    { set off [dbgetv $EndTime 0 $cnt $Values]
                     dbputv $Jdbsta 0 $cnt offdate $Entry($f)
                     dbputv $Jdbsta 0 $cnt sitechan.offdate $Entry($f)
                }
      sitechan.offdate  { set off [dbgetv $EndTime 0 $cnt $Values]
                     dbputv $Jdbsta 0 $cnt offdate $Entry($f)
                     dbputv $Jdbsta 0 $cnt sitechan.offdate $Entry($f)
                }

   }

}

proc PutBack {} {
   global EndTime
      set rec_cnt [dbquery $EndTime dbRECORD_COUNT]
   puts "rec_cnt in PutBack is $rec_cnt"
   #loop cnt 0 $rec_cnt {
   #}

}

proc CautionWin {} {
   global Pf f Entry Value list

   set w [uniqueW ""]
   set list ""
#   toplevel $w
#   frame $w.win1 -relief groove
#   frame $w.win2 -relief groove
#   pack $w.win1 -side top
#   pack $w.win2 -side bottom
#   set i 1
#   set j 1
  pfgetarr fields %${Pf}#field_map
#   label $w.win1.$j -text "Keep in mind \
#               that the following fields will also \
#               be changed: " \
#       -foreground blue
#   blt::table $w.win1 \
#      $w.win1.$j 0,$j -anchor w -fill x
#
#   incr j
#   set i 3
   for_array_keys changed fields {
      if {$Value == $changed} {
           set select $fields($changed)
           foreach one $select {
              lappend list $one
#             label $w.$j -relief ridge -text $one \
#                       -background lightskyblue
#             blt::table $w.win2 \
#                $w.$j $i,$j -anchor w -fill x
#             #incr i
#             incr j
           }
      }
   }

   caution .c {caution window} "$Value has been edited.\
          Keep in mind that the following fields will \
          also be changed:" $list
}

proc caution {w title msg args} {
    global Ok list
    catch {destroy $w}
    toplevel $w -class Dialog
    wm title $w $title
    wm iconname $w $title
puts "args is $args"

    # Create two frames in the main window. The top frame will hold the
    # message and the bottom one will hold the buttons.  Arrange them
    # one above the other, with any extra vertical space split between
    # them.

    frame $w.top -relief raised -border 1
    frame $w.bot -relief raised -border 1
    pack $w.top $w.bot -side top -fill both -expand yes

    # Create the message widget and arrange for it to be centered in the
    # top frame.

    message $w.top.msg -justify center \
            -font -Adobe-times-medium-r-normal--*-180* \
            -text $msg \
            -aspect 500 \
            -fg red
    pack $w.top.msg -side top -expand yes -padx 3 -pady 3

    # Create as many buttons as needed and arrange them from left to right
    # in the bottom frame.  Embed the left button in an additional sunken
    # frame to indicate that it is the default button, and arrange for that
    # button to be invoked as the default action for clicks and returns in
    # the dialog.

    if {[llength $args] > 0} {
        set arg [lindex $args 0]
        set i 1
        foreach arg $list {

            button $w.bot.$i -relief groove -text $arg \
                    -background lightskyblue \
                    -command "set Ok $i; destroy $w"
            pack $w.bot.$i -side left -expand yes
            incr i
        }
    }
    bind $w <Any-Enter> [list focus $w]
    focus $w

    grab set $w
    tkwait window $w
    grab release $w
    return $Ok
}

proc save_edit {} {
   global Mode w Entry Selection Old Selectioncol Db Record Expression
   global show_list New Selectionlb Selectionindex Expr change_values
   global j one type kind name dtype sid fname up comp sens trg pick calib
     #set field $Expression($name)
     set value $Entry($w)

     #set Expr($kind,$Selectioncol($w),$Selectionindex($w)) $Entry($w)
     $Selectionlb($w) delete $Selectionindex($w)
     $Selectionlb($w) insert $Selectionindex($w) $Entry($w)
}

proc cleanup {} {
   #exec sh -c "rm -r field"
   #exec sh -c "rmdir field"
   exit 1
}

########################################
# save value from entry window to table
# field value.
########################################

proc save_tmp {} {
   global Add fieldname cnt wanted count
   set one 1
   foreach each $wanted {
        set $each Add($fieldname($one))
        incr one
   }
}

proc RebuildDB {} {
   global Add fieldname cnt wanted count RIdbj5
}


#######################################
#Northridge gets special treatment.
#needs to know which stapar directory
```

```
#to look through.
########################################
proc northridge_out { network } {
        global Northridge pattern Ok net_name
        dialog .d {northridge stapars} {Which directory \
        would you like to proceed in?} {GEOS} {KINEMETRICS} \
        {SCEC} {SCSN}
        switch $Ok {

           0    { set pattern $network/$net_name/stapars/GEOS }

           1    { set pattern $network/$net_name/stapars/KINEMETRICS }

           2    { set pattern $network/$net_name/stapars/SCEC }

           3    { set pattern $network/$net_name/stapars/SCSN }

        }
}

################################################################
# displays selection window in order to select stapar fields
# and the order in which to view stapars
################################################################

proc stapar_fields {title} {
    global Fields N Sequence Checkbutton request
    global Possible Site Who DSerial DataLogger SampleRate \
        Comp SSerial Freq Damp Coil dbPreAmp dbGain Corner \
        Roll CountsV Highcorner HPR Theta Phi Model
    global Depth Latitude Longitude Elevation N_offset E_offset V_offset
    set w [uniqueW ""]
    toplevel $w
    wm title $w $title

    set maxrow 20
    set maxfields 10
    set Site "station"
    #set Who "up"
    set Who "is_it_running"
    set DSerial "dserial"
    set DataLogger "datalogger"
    set SampleRate "samplerate"
    set Comp "component"
    set Latitude "latitude"
    set Longitude "longitude"
    set Elevation "elevation"
    set N_offset "n-offset"
    set E_offset "e-offset"
    set V_offset "v-offset"
    set Model "sensor model"
    set Theta "verticle orientation"
    set Depth "emplacement_depth"
    set Phi "horizontal_orientation"
    set SSerial "sserial"
    set Freq "freq"
    set Coil "gen_const"
    set dbPreAmp "preamp"
    set dbGain "gain"
    set Corner "hf_corner"
    set Roll "hf_roll"
    set CountsV "digit_const"
    set Highcorner "hp_corner"
    set HPR "hp_roll"
    set lastrow [expr $maxrow+20]
    set span [expr $maxfields / $maxrow + 1]

    frame $w.f1
    button $w.f1.all \
        -text "all" \
        -command "check_all 1"
    button $w.f1.none \
        -text "none" \
        -command "check_all 0"

    pack $w.f1 -side top -fill x
    pack $w.f1.all $w.f1.none -side left -fill x -expand yes

    frame $w.f2
    button $w.f2.cancel \
        -text "cancel" \
        -command "set Ok 0 ; destroy $w"
    button $w.f2.ok \
        -text "ok" \
        -command "set Ok 1; destroy $w"
    pack $w.f2 -side bottom -fill x
    pack $w.f2.cancel $w.f2.ok -side left -fill x -expand yes

    set f $w.f3
    frame $f
    pack $f -side top -fill x

    if { [info exists Fields]} {
        unset Fields
        unset Sequence
        unset Checkbutton
        }

    set i 0
    set N 1
    set expr_row $maxrow
    set expr 0

    set Possible ""
########################################################
# This is the list of fields in stapars. later, this will
# be changed to be part of the parameter file.
#########################################################

    set fieldvals " $Site $Who sid latitude longitude elevation n_offset \
        e_offset v_offset $Comp type $SSerial theta phi $Freq $Damp $Coil \
        rsponsefile sens depth $DSerial $DataLogger $SampleRate filterresponsefiles \
        $dbPreAmp $dbGain $Corner $Roll $CountsV $Highcorner $HPR Time EndTime"

    foreach field $fieldvals {
puts "the field name is:   $field"


        set Fields($field) 0
        set Sequence($field) ""
        if { [is_expression $field] } {
            set row $expr_row
            incr expr_row
            set col 0
            set expr 1
        } else {
            set row [expr $i % $maxrow + 10]
            set col [expr 2*($i / $maxrow)]
            lappend Possible $field
            }
        label $f.l$i -textvariable Sequence($field) -width 2
        checkbutton $f.cb$i \
            -text $field \
            -anchor w \
            -relief flat \
            -variable Fields($field) \
            -command "fix_order \{$field\}"
        set Checkbutton($field) $f.cb$i
        blt::table $f $f.l$i  $row,$col -anchor e -fill x
        incr col
        if { $expr } {
            blt::table $f $f.cb$i $row,$col -columnspan 25 -anchor w -fill x
        } else {
            blt::table $f $f.cb$i $row,$col -anchor w -fill x
            }
        incr i
        }
puts "possible is : $Possible"
puts "the sequence is: $N"

    global Ok
    check_current $request

    update

    grab set $w
    tkwait window $w
    grab release $w
    global Ok wanted
    if { $Ok } {
        set wanted ""
        foreach field [array names Fields] {
            if { $Fields($field) } {
                lappend wanted $field
                }
            }
        set wanted [lsort -command by_value $wanted]
puts "new list has: $wanted"
        return $wanted
    } else {
        return ""
        }

}

########################################################################
# Arrange the fields of the table by default or by selection.
########################################################################

proc arrange { w } {
   global Request First_col default
   global request l locate cat

   set request [stapar_fields "Display order"]
puts "request is: $request"
   if { [llength $request] > 0 } {
        blt::busy hold $w
        set Request($w) $request

        set First_col($w) 0
        fix_columns $w
#        table_fill $w
```

```
        blt::busy forget $w

        }
}

###################################################################
# will start table window with default fields.
# default fields can be changed as needed
###################################################################

proc defaults { w } {
   global request First_col file_flag default
   global Site Who DataLogger dbGain Comp

   set Site "station"
   set Who "is_it_running"
   set DataLogger "datalogger"
   set dbGain "gain"
   set Comp "component"
   if {$file_flag != 0} {
        set request ""
        set default ""
        set fieldvals "{$Site} sid Time {$Comp} sens {$Who}"
        foreach field $fieldvals {
           lappend request $field
           lappend default $field
        }
   }

   if {$file_flag == 0} {
        set request ""
        set default ""
        set fieldvals "{$Site} sid {$Comp} sens rsponsefile \
                        {$DataLogger} {$dbGain}"
        foreach field $fieldvals {
           lappend request $field
           lappend default $field
        }
   }

   set First_col($w) 0
   fix_columns $w
}

#
# The field selection windows
#

proc fix_order { field } {
    global Fields N Sequence
    if { $Fields($field) } {
        set Sequence($field) $N
        incr N
    } else {
        set old $Sequence($field)
        set Sequence($field) ""
        foreach f [array names Sequence] {
            if { $Sequence($f) > $old } {
                set Sequence($f) [expr $Sequence($f)-1]
                }
            }
        incr N -1
        }
    }
```

```
########################################
# When all fields of stapars are to be
# seen.
########################################

proc check_all { flag } {
    global Possible Sequence Fields N Checkbutton
    set N 1
    foreach field $Possible {
        set Sequence($field) ""
        $Checkbutton($field) deselect
        }
    if { $flag } {
        foreach field $Possible {
            set Sequence($field) $N
            incr N
            $Checkbutton($field) select
            }
        }
    }


proc by_value { a b } {
    global Sequence
    return [expr $Sequence($a)-$Sequence($b)]
    }

proc check_current { request } {
    global Sequence Checkbutton
    foreach field $request {
        if { [info exists Sequence($field)] } {
            set Sequence($field) 0
            $Checkbutton($field) invoke
        }
    }
}

proc is_expression { s } {
    return [expr ! [regexp {^[a-zA-Z_.0-9]*$} $s] ]
    }

proc dialog {w title msg args} {
    global Ok
    catch {destroy $w}
    toplevel $w -class Dialog
    wm title $w $title
    wm iconname $w $title

    # Create two frames in the main window. The top frame will hold the
    # message and the bottom one will hold the buttons.  Arrange them
    # one above the other, with any extra vertical space split between
    # them.

    frame $w.top -relief raised -border 1
    frame $w.bot -relief raised -border 1
    pack $w.top $w.bot -side top -fill both -expand yes

    # Create the message widget and arrange for it to be centered in the
    # top frame.

    message $w.top.msg -justify center \
            -font -Adobe-times-medium-r-normal--*-180* \
            -text $msg \
            -aspect 500 \
            -fg red
    pack $w.top.msg -side top -expand yes -padx 3 -pady 3

    # Create as many buttons as needed and arrange them from left to right
    # in the bottom frame.  Embed the left button in an additional sunken
    # frame to indicate that it is the default button, and arrange for that
    # button to be invoked as the default action for clicks and returns in
    # the dialog.

    if {[llength $args] > 0} {
        set arg [lindex $args 0]
        frame $w.bot.0 -relief sunken -border 1
        pack $w.bot.0 -side left -expand yes -padx 10 -pady 10
        button $w.bot.0.button -text [lindex $arg 0] \
                -command "set Ok 0; destroy $w"
        pack $w.bot.0.button -expand yes -padx 6 -pady 6
        bind $w <Return> "set Ok 0; destroy $w"
        focus $w

        set i 1
        foreach arg [lrange $args 1 end] {
            button $w.bot.$i -text [lindex $arg 0] \
                    -command "set Ok $i; destroy $w"
            pack $w.bot.$i -side left -expand yes -padx 10
            incr i
        }
    }
    bind $w <Any-Enter> [list focus $w]
    focus $w

    grab set $w
    tkwait window $w
    grab release $w
    return $Ok
}

#####################################
# This is the main Field2db window.
#####################################
proc menu_setup { p } {
    global Readable_epoch_time \
        Pf \
        Hide_null_columns \
        Hide_duplicate_columns \
        Font  new_net
    set w $p.mb
    frame $w

    menubutton $w.file -text File -underline 0 -menu $w.file.m
    menu $w.file.m
    $w.file.m add command -label "Build New Table" -underline 0 \
            -command "create_table"

    $w.file.m add command -label "Quit field2db" -underline 0 \
            -command "destroy ."

        menubutton $w.view -text View -underline 0 -menu $w.view.m
        menu $w.view.m
        $w.view.m add command -label "Arrange" -underline 0 -command "arrange $p"
        #$w.view.m add command -label "Sort" -underline 0 -command "uc"
        #$w.view.m add command -label "Subset" -underline 1 -command "uc"

    menubutton $w.help -text Help -underline 0 -menu $w.help.m
    menu $w.help.m
    global Major Minor Date
    $w.help.m add command -label "About field2db"  -command "uc"

    pack $w.file -side left
    pack $w.view -side left
    pack $w.help -side right

    tk_menuBar $w $w.file $w.view $w.help

    return $w

    }

########################################################################
#  Procedure to scan the lines of the location section of a stapar file.
#  Also, it will count the number of stations located in this section
#  and set that number as num_sta.
#
#  NOTE:   The output will be written to standard output for now!
#
########################################################################
# LOCATION BLOCK INPUT
######################

proc loc_block_in {$line $i $t($time)} {
   global line i stapar t check num_sta site who sid lat long elev \
        noff eoff voff num stacode file_cnt sta_cnt strg_cnt count \
        time sta_file new new_file cnt
   global sitename tabletime time

   set field_cnt [expr [llength $line] -8]
   incr i
   incr cnt
   incr num_sta($time)
   scan $line "%s %s %s %10f %10f %10f %8f %8f %8f" \
              site($i,$t($time)) who($i,$t($time))\
              sid($i,$t($time)) lat($i,$t($time))\
              long($i,$t($time)) elev($i,$t($time))\
              noff($i,$t($time)) eoff($i,$t($time))\
              voff($i,$t($time))

   set strg_cnt [string length $site($i,$t($time))]
   set sta_cnt($i,$t($time)) $strg_cnt
   if {$stacode == 1} {
      set sitename($i) -
   }
}

########################################################################
#  Procedure to scan for the number lines in the sensor section for
#  each station.  If the number of fields for any linein the section
#  is twelve (where the station names are located) we will set a
#  chan_cnt to count the number of channels for each station.
#########################################################################
#  SENSOR INPUT
###############

proc sensor_in {$line $i $j $t($time)} {
   global line i j num_sta site numfield comp type s_serial theta phi \
        freq damp coil rfile sens depth chan_cnt t time ok stacode

   ############################
   #scan the correct line length
   if {$stacode == 1} {
      set ok($i,$j,$time) 0
   }
   switch $numfield {

      12 {    set chan_cnt($i) $j
              set j 1
              incr i
              scan $line "%s %4d %4s %6s %5d %3d %8f %8f %8f %10s %4s \
                        %5d" site($i,$t($time) comp($i,$j,$t($time))\
                        type($i,$j,$t($time)) s_serial($i,$j,$t($time)) \
                        theta($i,$j,$t($time)) phi($i,$j,$t($time))\
                        freq($i,$j,$t($time)) damp($i,$j,$t($time))\
                        coil($i,$j,$t($time)) rfile($i,$j,$t($time))\
                        sens($i,$j,$t($time)) depth($i,$j,$t($time))
         }
      11 {    incr j
              scan $line "%4d %4s %6s %5d %3d %8f %8f %8f %10s %4s %5d" \
                         comp($i,$j,$t($time)) type($i,$j,$t($time))\
                         s_serial($i,$j,$t($time)) theta($i,$j,$t($time))\
                         phi($i,$j,$t($time)) freq($i,$j,$t($time))\
                         damp($i,$j,$t($time)) coil($i,$j,$t($time))\
                         rfile($i,$j,$t($time)) sens($i,$j,$t($time))\
                         depth($i,$j,$t($time))
         }

   }
}

##########################################################################
#  Procedure to scan for the lines of the amplifier description section.
#  This procedure is the same as sensor_in , by setting a chan_cnt to
#  count the number of channels for each station.  Each time that the
#  number of fields for the line is 9 the chan_cnt is set.
##########################################################################
#  AMP_IN
##########

proc amp_in {$line $i $k $t($time)} {
   global line i k num_sta sid comp preamp gain corner roll cv\
        hcorner hpr numfield look chan_cnt t time

   ##############################
   #scan the correct line length

   switch $numfield {

      9 {  set k 1
           incr i
           scan $line "%4d %4d %8f %8f %6f %4d %8f %10f %4d"\
                sid($i,$k,$t($time)) comp($i,$k,$t($time))\
                preamp($i,$k,$t($time)) gain($i,$k,$t($time))\
                corner($i,$k,$t($time)) roll($i,$k,$t($time))\
                cv($i,$k,$t($time)) hcorner($i,$k,$t($time))\
                hpr($i,$k,$t($time))
        }
      8 {  incr k
           scan $line "%4d %8f %8f %6f %4d %8f %10f %4d"\
                comp($i,$k,$t($time)) preamp($i,$k,$t($time))\
                gain($i,$k,$t($time)) corner($i,$k,$t($time))\
                roll($i,$k,$t($time)) cv($i,$k,$t($time))\
                hcorner($i,$k,$t($time)) hpr($i,$k,$t($time))
        }
   }
```

```
}

########################################################################
#  Procedure to scan for the lines of the network section.
########################################################################
#  NET_IN
#########
proc net_in {$line $n $t($time)} {
   global line n n_type n_label buff_siz b_off tag_off tag_siz \
        w_siz npts s_time t time

   incr n
   scan $line "%8s %9s %8d %6d %6d %6d %6d %4d %s" \
                n_type($n,$t($time)) n_label($n,$t($time)) \
                buff_siz($n,$t($time)) b_off($n,$t($time)) \
                tag_off($n,$t($time)) tag_siz($n,$t($time))\
                w_siz($n,$t($time)) npts($n,$t($time)) \
                s_time($n,$t($time))

}

########################################################################
#  Procedure to scan for the datalogger fields of the stapar file.
#  The last field, FilterResponseFiles will have to be handled
#  as a multidimensional array.
########################################################################
#  LOGGER_IN
############

proc logger_in  {$line $i $j $x $t($time)} {
   global line i j x sid serial datalog s_rate fr_files dl_flag \
        resp_file_cnt t time

   set dl_flag [expr ([llength $line] -4)]
   set x $dl_flag
   set j 1
   incr i

   ##########################################################
   #each station will not always have the same number of
   #FilterResponseFiles. This switch works much like
   #chan_cnt does in sensor_in and amp_in, it sets a specific
   #number of FilterResponseFiles for each station.
   switch $dl_flag {

      1 {set resp_file_cnt($i,$t($time)) $dl_flag
         scan $line "%4d %6s %10s %10f %s"\
         sid($i,$t($time)) serial($i,$j,$t($time)) \
                datalog($i,$j,$t($time)) s_rate($i,$j,$t($time)) \
                fr_files($i,1,$t($time))}

      2 {set resp_file_cnt($i,$t($time)) $dl_flag
         scan $line "%4d %6s %10s %10f %s %s"\
         sid($i,$t($time)) serial($i,$j,$t($time)) \
                datalog($i,$j,$t($time)) s_rate($i,$j,$t($time)) \
                fr_files($i,1,$t($time)) fr_files($i,2,$t($time))}

      3 {set resp_file_cnt($i,$t($time)) $dl_flag
         scan $line "%4d %6s %10s %10f %s %s %s"\
         sid($i,$t($time)) serial($i,$j,$t($time)) \
                datalog($i,$j,$t($time)) s_rate($i,$j,$t($time)) \
                fr_files($i,1,$t($time)) fr_files($i,2,$t($time))\
                fr_files($i,2,$t($time)) fr_files($i,3,$t($time))}
```

```
    4 {set resp_file_cnt($i,$t($time)) $dl_flag
       scan $line "%4d %6s %10s %10f %s %s %s %s"\
       sid($i,$t($time)) serial($i,$j,$t($time)) \
               datalog($i,$j,$t($time)) s_rate($i,$j,$t($time)) \
               fr_files($i,1,$t($time)) fr_files($i,2,$t($time)) \
               fr_files($i,3,$t($time)) fr_files($i,4,$t($time))}

    5 {set resp_file_cnt($i,$t($time)) $dl_flag
       scan $line "%4d %6s %10s %10f %s %s %s %s %s"\
               sid($i,$t($time)) serial($i,$j,$t($time))  \
               datalog($i,$j,$t($time)) s_rate($i,$j,$t($time)) \
               s_rate($i,$j,$t($time)) fr_files($i,1,$t($time))\
               fr_files($i,2,$t($time)) fr_files($i,3,$t($time))\
               fr_files($i,4,$t($time)) fr_files($i,5,$t($time))}

    6 {set resp_file_cnt($i,$t($time)) $dl_flag
       scan $line "%4d %6s %10s %10f %s %s %s %s %s %s"\
               sid($i,$t($time)) serial($i,$j,$t($time)) \
               datalog($i,$j,$t($time)) s_rate($i,$j,$t($time)) \
               fr_files($i,1,$t($time)) fr_files($i,2,$t($time)) \
               fr_files($i,3,$t($time)) fr_files($i,4,$t($time)) \
               fr_files($i,5,$t($time)) fr_files($i,6,$t($time))}

    7 {set resp_filge_cnt($i,$t($time)) $dl_flag
       scan $line "%4d %6s %10s %10f %s %s %s %s %s %s %s"\
       sid($i,$t($time)) serial($i,$j,$t($time)) \
               datalog($i,$j,$t($time)) s_rate($i,$j,$t($time)) \
               fr_files($i,1,$t($time)) fr_files($i,2,$t($time)) \
               fr_files($i,3,$t($time)) fr_files($i,4,$t($time)) \
               fr_files($i,5,$t($time)) fr_files($i,6,$t($time)) \
               fr_files($i,7,$t($time))}

    8 {set resp_file_cnt($i,$t($time)) $dl_flag
       scan $line "%4d %6s %10s %10f %s %s %s %s %s %s %s %s"\
       sid($i,$t($time)) serial($i,$j,$t($time)) \
               datalog($i,$j,$t($time)) s_rate($i,$j,$t($time)) \
               fr_files($i,1,$t($time)) fr_files($i,2,$t($time)) \
               fr_files($i,3,$t($time)) fr_files($i,4,$t($time)) \
               fr_files($i,5,$t($time)) fr_files($i,6,$t($time)) \
               fr_files($i,7,$t($time)) fr_files($i,8,$t($time))}

    9 {set resp_file_cnt($i,$t($time)) $dl_flag
       scan $line "%4d %6s %10s %10f %s %s %s %s %s %s %s %s %s"\
       sid($i,$t($time)) serial($i,$j,$t($time)) \
               datalog($i,$j,$t($time)) s_rate($i,$j,$t($time)) \
               fr_files($i,1,$t($time)) fr_files($i,2,$t($time)) \
               fr_files($i,3,$t($time)) fr_files($i,4,$t($time)) \
               fr_files($i,5,$t($time)) fr_files($i,6,$t($time)) \
               fr_files($i,7,$t($time)) fr_files($i,8,$t($time)) \
               fr_files($i,9,$t($time))}

   10 {set resp_file_cnt($i,$t($time)) $dl_flag
       scan $line "%4d %6s %10s %10f %s %s %s %s %s %s %s %s %s %s"\
       sid($i,$t($time)) serial($i,$j,$t($time)) \
               datalog($i,$j,$t($time)) s_rate($i,$j,$t($time)) \
               fr_files($i,1,$t($time)) fr_files($i,2,$t($time)) \
               fr_files($i,3,$t($time)) fr_files($i,4,$t($time)) \
               fr_files($i,5,$t($time)) fr_files($i,6,$t($time)) \
               fr_files($i,7,$t($time)) fr_files($i,8,$t($time)) \
               fr_files($i,9,$t($time)) fr_files($i,10,$t($time))}

   default {puts "ERROR!  No more space!"}

}
```

```
}

##########################################################################
#  Procedure to output the first section of a stapar file. It will
#  output to standard output or to a file. All character strings will
#  be left_justified and Tcl automatically right-justifies all numbers.
##########################################################################
#  LOCATION BLOCK OUTPUT
########################

proc loc_block_out {$sta_file($time) $site($i,$t($time)) \
        $who($i,$t($time)) $sid($i,$t($time)) $lat($i,$t($time)) \
        $long($i,$t($time)) $elev($i,$t($time)) $noff($i,$time)) \
        $eoff($i,$t($time)) $voff($i,$t($time))} {

   global i num_sta sta_cnt strg_cnt site who sid lat long elev \
        noff eoff voff t file_cnt file_id time new sta_file \
        new_file cnt

   for {set i 1} {$i <= $num_sta($time)} {incr i} {
      set sta_cnt($i,$t($time)) [string length $site($i,$t($time))]
      if {$sta_cnt($i,$t($time)) > 4} {
         puts $sta_file($time) [format "%s %s %4d %10.5f %10.5f \
                %10.3f %8.1f %8.1f %8.1f" \
                $site($i,$t($time)) $who($i,$t($time)) \
                $sid($i,$t($time)) $lat($i,$t($time)) \
                $long($i,$t($time)) $elev($i,$t($time)) \
                $noff($i,$t($time)) $eoff($i,$t($time)) \
                $voff($i,$t($time))]
         continue
      } else {
         puts $sta_file($time) [format "%-4s %-s %4d %10.5f %10.5f \
                   %10.3f %8.1f %8.1f %8.1f" \
                $site($i,$t($time)) $who($i,$t($time)) $sid($i,$t($time))\
                $lat($i,$t($time)) $long($i,$t($time))\
                $elev($i,$t($time)) $noff($i,$t($time))\
                $eoff($i,$t($time)) $voff($i,$t($time))]
      }

   }
}

#########################################################################
#  Procedure to output the sensor section of a stapar file.
#########################################################################
#  SENSOR OUTPUT
################

proc sensor_out {$site($i,$t($time)) $comp($i,$j,$t($time))\
        $type($i,$j,$t($time)) $s_serial($i,$j,$t($time))\
        $theta($i,$j,$t($time)) $phi($i,$j,$t($time)) \
        $freq($i,$j,$t($time)) $damp($i,$j,$t($time)) \
        $coil($i,$j,$t($time)) $rfile($i,$j,$t($time)) \
        $sens($i,$j,$t($time)) $depth($i,$j,$t($time))} {

   global i j num_sta chan_cnt site comp type s_serial theta \
        phi freq damp coil rfile sta_cnt strg_cnt sens depth time \
        time t sta_file rt

   for {set i 1} {$i <= $num_sta($time)} {incr i} {
      set sta_cnt($i,$t($time)) [string length $site($i,$t($time))]
      puts "numsta in sensor_out is :$num_sta($time)"
      puts "what's sta_cnt($i,$t($time)): $sta_cnt($i,$t($time))"
      switch $sta_cnt($i,$t($time)) {
        5 {set j 1
           puts $sta_file($time) [format "%s %3d %-4s %6s %5d %3d \
                %8.3f %8.3f %8.4f %-11s %-4s %5d" \
                $site($i,$t($time)) $comp($i,$j,$t($time)) \
                $type($i,$j,$t($time)) $s_serial($i,$j,$t($time)) \
                $theta($i,$j,$t($time)) $phi($i,$j,$t($time))\
                $freq($i,$j,$t($time)) $damp($i,$j,$t($time))\
                $coil($i,$j,$t($time)) $rfile($i,$j,$t($time))\
                $sens($i,$j,$t($time)) $depth($i,$j,$t($time))]
                for {set j 2} {$j <= $chan_cnt($i)} {incr j} {
                    puts $sta_file($time) [format "%9d %-4s %6s %5d \
                         %3d %8.3f %8.3f %8.4f %-11s %-4s %5d" \
                         $comp($i,$j,$t($time)) $type($i,$j,$t($time))\
                         $s_serial($i,$j,$t($time))\
                         $theta($i,$j,$t($time)) $phi($i,$j,$t($time))\
                         $freq($i,$j,$t($time)) $damp($i,$j,$t($time))\
                         $coil($i,$j,$t($time)) $rfile($i,$j,$t($time))\
                         $sens($i,$j,$t($time)) $depth($i,$j,$t($time))]
                }
           }
        6 {set j 1
           puts $sta_file($time) [format "%s %2d %-4s %6s %5d %3d  \
                %8.3f %8.3f %8.4f %-11s %-4s %5d" \
                $site($i,$t($time)) $comp($i,$j,$t($time)) \
                $type($i,$j,$t($time)) $s_serial($i,$j,$t($time)) \
                $theta($i,$j,$t($time)) $phi($i,$j,$t($time))\
                $freq($i,$j,$t($time)) $damp($i,$j,$t($time))\
                $coil($i,$j,$t($time)) $rfile($i,$j,$t($time))\
                $sens($i,$j,$t($time)) $depth($i,$j,$t($time))]
                for {set j 2} {$j <= $chan_cnt($i)} {incr j} {
                   puts $sta_file($time) [format "%9d %-4s %6s %5d \
                         %3d %8.3f %8.3f %8.4f %-11s %-4s %5d" \
                         $comp($i,$j,$t($time)) $type($i,$j,$t($time))\
                         $s_serial($i,$j,$t($time))\
                         $theta($i,$j,$t($time)) $phi($i,$j,$t($time))\
                         $freq($i,$j,$t($time)) $damp($i,$j,$t($time))\
                         $coil($i,$j,$t($time)) $rfile($i,$j,$t($time))\
                         $sens($i,$j,$t($time)) $depth($i,$j,$t($time))]
                }
           }
        default {set j 1
                  puts $sta_file($time) [format "%-4s %4d %-4s %6s %5d \
                         %3d %8.3f %8.3f %8.4f %-11s %-4s %5d" \
                         $site($i,$t($time)) $comp($i,$j,$t($time)) \
                         $type($i,$j,$t($time)) \
                            $s_serial($i,$j,$t($time))\
                            $theta($i,$j,$t($time)) $phi($i,$j,$t($time))\
                            $freq($i,$j,$t($time)) $damp($i,$j,$t($time))\
                            $coil($i,$j,$t($time)) $rfile($i,$j,$t($time))\
                            $sens($i,$j,$t($time)) $depth($i,$j,$t($time))]
                            for {set j 2} {$j <= $chan_cnt($i)} {incr j} {
                              puts $sta_file($time) [format "%9d %-4s %6s %5d %3d %8.3f %8.
3f %8.4f\
                                   %-11s %-4s %5d"\
                            $comp($i,$j,$t($time)) $type($i,$j,$t($time))\
                            $s_serial($i,$j,$t($time))\
                            $theta($i,$j,$t($time)) $phi($i,$j,$t($time))\
                            $freq($i,$j,$t($time)) $damp($i,$j,$t($time))\
                            $coil($i,$j,$t($time)) $rfile($i,$j,$t($time))\
                            $sens($i,$j,$t($time)) $depth($i,$j,$t($time))]
                           }
                         }
                }
            }
        }

   ########################################################################
   #  Procedure to output the datalogger section.
   ########################################################################
   #  LOOGER_OUT
   #############

   proc logger_out {$sta_file($time) $sid($i,$t($time)) \
           $serial($i,$j,$t($time)) $datalog($i,$j,$t($time))\
           $s_rate($i,$j,$t($time)) $fr_files($i,$x,$t($time))} {
      global i j sid serial datalog s_rate fr_files dl_flag \
           num_sta x resp_file_cnt sta_file t time
      set j 1
      for {set i 1} {$i <= $num_sta($time)} {incr i} {
         switch $resp_file_cnt($i,$t($time)) {
            0    { puts $sta_file($time) [format "%4d %6s %-10s \
                      %10.4f %s" $sid($i,$t($time))\
                      $serial($i,$j,$t($time)) $datalog($i,$j,$t($time))\
                      $s_rate($i,$j,$t($time)) $fr_files($i,1,$t($time))]
                 }
            2    { puts $sta_file($time) [format "%4d %6s %-10s \
                      %10.4f %s %s" $sid($i,$t($time)) \
                      $serial($i,$j,$t($time)) $datalog($i,$j,$t($time))\
                      $s_rate($i,$j,$t($time)) $fr_files($i,1,$t($time))\
                      $fr_files($i,2,$t($time))]
                 }
            3    { puts $sta_file($time) [format "%4d %6s %-10s \
                      %10.4f %s %s %s" $sid($i,$t($time))\
                      $serial($i,$j,$t($time)) $datalog($i,$j,$t($time))\
                      $s_rate($i,$j,$t($time)) $fr_files($i,1,$t($time))\
                      $fr_files($i,2,$t($time)) $fr_files($i,3,$t($time))]
                 }
            4    { puts $sta_file($time) [format "%4d %6s %-10s \
                      %10.4f %s %s %s %s" $sid($i,$t($time))\
                      $serial($i,$j,$t($time)) $datalog($i,$j,$t($time))\
                      $s_rate($i,$j,$t($time)) $fr_files($i,1,$t($time))\
                      $fr_files($i,2,$t($time)) $fr_files($i,3,$t($time))\
                      $fr_files($i,4,$t($time))]
                 }
            5    { puts $sta_file($time) format "%4d %6s %-10s \
                      %10.4f %s %s %s %s %s" $sid($i,$t($time))\
                      $serial($i,$j,$t($time)) $datalog($i,$j,$t($time))\
                      $s_rate($i,$j,$t($time)) $fr_files($i,1,$t($time))\
                      $fr_files($i,2,$t($time)) $fr_files($i,3,$t($time))\
                      $fr_files($i,4,$t($time)) $fr_files($i,5,$t($time))]
                 }
            6    { puts $sta_file($time) [format "%4d %6s %-10s \
                      %10.4f %s %s %s %s %s %s" $sid($i,$t($time))\
                      $serial($i,$j,$t($time)) $datalog($i,$j,$t($time))\
                      $s_rate($i,$j,$t($time)) $fr_files($i,1,$t($time))\
                      $fr_files($i,2,$t($time)) $fr_files($i,3,$t($time))\
                      $fr_files($i,4,$t($time)) $fr_files($i,5,$t($time))\
                      $fr_files($i,6,$t($time))]
                 }
            7    { puts $sta_file($time) [format "%4d %6s %-10s \
                      %10.4f %s %s %s %s %s %s %s"  $sid($i,$t($time))\
                      $serial($i,$j,$t($time)) $datalog($i,$j,$t($time))\
                      $s_rate($i,$j,$t($time)) $fr_files($i,1,$t($time))\
                      $fr_files($i,2,$t($time)) $fr_files($i,3,$t($time))\
                      $fr_files($i,4,$t($time)) $fr_files($i,5,$t($time))\
                      $fr_files($i,6,$t($time)) $fr_files($i,7,$t($time))]
                 }
            8    { puts $sta_file($time) [format "%4d %6s %-10s \
                %10.4f %s %s %s %s %s %s %s %s" $sid($i,$t($time)) \
                $serial($i,$j,$t($time)) $datalog($i,$j,$t($time))\
                $s_rate($i,$j,$t($time)) $fr_files($i,1,$t($time))\
                $fr_files($i,2,$t($time)) $fr_files($i,3,$t($time))\
                $fr_files($i,4,$t($time)) $fr_files($i,5,$t($time))\
                $fr_files($i,6,$t($time)) $fr_files($i,7,$t($time))\
                $fr_files($i,8,$t($time))]
            }
        9   { puts $sta_file($time) [format "%4d %6s %-10s \
                %10.4f %s %s %s %s %s %s %s %s %s" $sid($i,$t($time)) \
                $serial($i,$j,$t($time)) $datalog($i,$j,$t($time))\
                $s_rate($i,$j,$t($time)) $fr_files($i,1,$t($time))\
                $fr_files($i,2,$t($time)) $fr_files($i,3,$t($time))\
                $fr_files($i,4,$t($time)) $fr_files($i,5,$t($time))\
                $fr_files($i,6,$t($time)) $fr_files($i,7,$t($time))\
                $fr_files($i,8,$t($time)) $fr_files($i,9,$t($time))]
            }
       10   { puts $sta_file($time) [format "%4d %6s %-10s %10.4f \
                %s %s %s %s %s %s %s %s %s %s" $sid($i,$t($time)) \
                $serial($i,$j,$t($time)) $datalog($i,$j,$t($time))\
                $s_rate($i,$j,$t($time)) $fr_files($i,1,$t($time))\
                $fr_files($i,2,$t($time)) $fr_files($i,3,$t($time))\
                $fr_files($i,4,$t($time)) $fr_files($i,5,$t($time))\
                $fr_files($i,6,$t($time)) $fr_files($i,7,$t($time))\
                $fr_files($i,8,$t($time)) $fr_files($i,9,$t($time))\
                $fr_files($i,10,$t($time))]
            }
        }
      }
}

###################################################################
#   Procedure to output the amplifier description section.
###################################################################
#   AMP_OUT
###########

proc amp_out {$sid($i,$t($time)) $comp($i,$k,$t($time)) \
        $preamp($i,$k,$t($time)) $gain($i,$k,$t($time)) \
        $corner($i,$k,$t($time)) $roll($i,$k,$t($time)) \
        $cv($i,$k,$t($time)) $hcorner($i,$k,$t($time)) \
        $hpr($i,$k,$t($time)) $sta_file($time)} {

   global i k num_sta chan_cnt sid comp preamp gain corner \
        roll cv hcorner hpr t time sta_file

   for {set i 1} {$i <= $num_sta($time)} {incr i} {
      set k 1
      puts $sta_file($time) [format "%4d %4d %8.4f %8.3f %6.2f %4d %8.1f \
                %10.4f %2.2f" $sid($i,$t($time)) $comp($i,$k,$t($time)) \
                $preamp($i,$k,$t($time)) $gain($i,$k,$t($time)) \
                $corner($i,$k,$t($time)) $roll($i,$k,$t($time)) \
                $cv($i,$k,$t($time)) $hcorner($i,$k,$t($time)) \
                $hpr($i,$k,$t($time))]
      for {set k 2} {$k <= $chan_cnt($i)} {incr k} {
         puts $sta_file($time) [format "%9d %8.4f %8.3f %6.2f %4d %8.1f\
                %10.4f %2.2f" $comp($i,$k,$t($time)) \
                $preamp($i,$k,$t($time)) $gain($i,$k,$t($time)) \
                $corner($i,$k,$t($time)) $roll($i,$k,$t($time)) \
                $cv($i,$k,$t($time)) $hcorner($i,$k,$t($time)) \
                $hpr($i,$k,$t($time))]
      }
   }
}
```

```
##########################################################################
#   Procedure to output the network section.
##########################################################################
#   NET_OUT
###########

proc net_out {$sta_file($time) $n_type($n,$t($time)) \
   $n_label($n,$t($time)) $buff_siz($n,$t($time)) $b_off($n,$t($time)) \
   $tag_off($n,$t($time)) $tag_siz($n,$t($time)) $w_siz($n,$t($time)) \
   $npts($n,$t($time)) $s_time($n,$t($time))} {

   global n num_sta n_type n_label buff_siz b_off tag_off tag_siz \
        w_siz npts s_time t time sta_file
   set n 1
   puts $sta_file($time) [format "%-8s %-9s %8d %6d %6d %6d %6d %4d \
        %s" $n_type(1,$t($time)) $n_label(1,$t($time)) \
        $buff_siz(1,$t($time)) $b_off(1,$t($time)) $tag_off(1,$t($time)) \
        $tag_siz(1,$t($time)) $w_siz(1,$t($time)) $npts(1,$t($time)) \
        $s_time(1,$t($time))]
    }

###############################################################
#set some variables and set array to keep track of file or
#files in a directory.
###############################################################
proc all {} {
   global pattern i j k t time site comp rt type serial theta \
        phi freq damp coil rfile sens depth file_flag value
   global new_net rt new_time files new file_cnt time pattern\
        file_flag site i t
        set pattern $pattern
        set Wcnt 0
        set file_flag 1
        set tcl_precision 17
        set sec_arg $pattern
        set dir $pattern
        set dir_flag $pattern
        start
}

##################################################################
# This procedure opens the selected stapar file or files.
##############################################################

proc start {} {
   global i time line files j k n dir sec_arg cnt num_sta \
        t strg_cnt numfield dl_flag rt file_cnt pattern \
        site comp rt type serial theta phi freq damp coil \
        rfile sens depth sta_file chan_cnt file_flag new_file \
        ep year epoch_clk new s_serial clock new_time new_net_flag

   if {$file_flag != 0} {
      puts "the pattern is: $pattern"
      set code [catch {exec sh -c "ls $pattern/*S "} files]
      if {$code == 1} {

         puts "error! $files, exiting ..."
         exit
      }
   }
   if {$file_flag == 0} {

      if {$new_net_flag > 0} {
```

```
      set value "$pattern/stapars/$rt($time)"
      set files $value
      puts "the rt is: $rt($time)"
   } else {
      set value "$pattern/$clock"
      set now [file tail $value]
      set files $value
      puts "the new is: $clock"
   }

   puts "the value is: $value"
   #puts "now is: $now"


}

set sec_arg 0
set i 1
set time 1
set file_cnt 1

###########################################
# opens the stapar file/files and checks
# to see if it is a stapar file.
###########################################

foreach stapar $files {
   puts "what's stapar: $stapar"
   set file_id($time) $stapar
   set t($time) $stapar
   set count($stapar) $file_cnt
if {$file_flag == 0} {
   set find [string length $value]
   set length [expr ($find - 13)]
   set new [file tail $stapar]
   set new_file($time) NEW.$new
   set rt($time) $new
   set modify [string trimleft [string range $rt($time) 0 4]]
   set ep [epoch "19$modify"]
   set date [strtime "$ep"]
   set year [yearday "$ep"]
   set epoch_clk($time) "($year) $date"

   ########################################
   #is file a stapar file?

   set open_id2($time) [open $stapar r]
   gets $open_id2($time) line
   set check1 [lindex $line 0]
   set check2 [lindex $line 2]
   if {($check1 != "Site") && ($check2 != "_SID")} {
      puts "$file_flag not a stapar file, exiting ....."
      exit
      incr file_cnt
      incr time
      continue
   }
   if {$file_flag != 0} {

      set find [string length $pattern]
      set found [string trimleft [string range $stapar $find end]]
      set new [file tail $stapar]
      set new_file($time) NEW.$new
      set rt($time) $new
```

```
        set modify [string trimleft [string range $rt($time) 0 4]]
        set ep [epoch "19$modify"]
        set date [strtime "$ep"]
        set year [yearday "$ep"]
        set epoch_clk($time) "($year) $date"
    }
    ########################################
    #is file a stapar file?

    set err [catch {open $stapar r} open_id2($time)]
    if {$err == 1} {

       puts "error! $open_id2($time), exiting ..."
       exit
    }
    gets $open_id2($time) line
    set check1 [lindex $line 0]
    set check2 [lindex $line 2]
    if {($check1 != "Site") && ($check2 != "_SID")} {
       puts "File not a stapar file, exiting ....."
       exit
    }
    close $open_id2($time)
    incr file_cnt
    incr time
  }
######################################
#initialize some counters and flags.
######################################

  set j 0
  set x 0
  set k 0
  set cnt 0
  set id_flag 0
  set file_cnt 1
  set time 1
  set num_sta($time) $cnt

###################################################
#read the lines in from the stapar file
#and count the number of fields that it contains.
#if the count is valid, start parsing data from
#stapar for output.
###################################################

  foreach spe $files {
     set open_id2($time) [open $spe r]
     puts "files opened: $file_cnt"
     while {[gets $open_id2($time) line] >= 0} {
        set numfield [llength $line]
        set line_indx1 [lindex $line 0]
        set line_indx2 [lindex $line 2]
        if {($line_indx1 == "Site") || ($line_indx1 == "_SID") \
           || ($line_indx1 == "Net_Type")} {
              set id_flag 0
        }
####################################################################
# matches the id_flag up with the right section in the stapar
# in order to scan correctly.
# in order to set the right flag for the right section,
# we have to check the section headers
# line_indx1 and indx2 represent key names in section header lines.
####################################################################
```

```
        switch $id_flag {

           loc     {loc_block_in line i t($time)}
           sen     {sensor_in line i j t($time)}
           d_log   {logger_in line i j x t($time)}
           amp     {amp_in line i k t($time)}
           net     {net_in line n t($time)}

           default {if {($line_indx1 != "Site") || ($line_indx1 != "_SID") \
                            || ($line_indx1 != "Net_Type")} {
              if {($numfield == 9) && ($line_indx2 == "_SID")} {
                 set i 0
                 set cnt 0
                 set num_sta($time) $cnt
                 set id_flag loc
              } elseif  {($numfield == 12)} {
                 set id_flag sen
                 set i 0
              } elseif  {($numfield == 5)} {
                 set chan_cnt($i) $j
                 set id_flag d_log
                 set i 0
                 set x 0
              } elseif  {($numfield == 9) && ($line_indx2 == "dbPreAmp")} {
                 set id_flag amp
                 set resp_file_cnt($i,$t($time)) $dl_flag
                 set i 0
                 set x 0
              } elseif  {($numfield == 9) && ($line_indx2 == "Buf_Size")} {
                 set id_flag net
                 set chan_cnt($i) $k
                 set n 0
              } else   {
                 set id_flag 0
              }
              }
           }
        }
     }
   }
   close $open_id2($time)
   incr file_cnt
   incr time
   }
   set time 1
   if {$file_flag == 0} {show_table}
   if {$file_flag != 0} {show_table}
}

########################################################################
# Write section headers and each section's output to new stapar files
########################################################################

proc write_field {} {
   global time t site who sid lat long elev noff eoff voff i Add comp sens \
      type serial theta phi freq damp coil rfile depth j new_time new_comp \
      preamp gain corner roll cv hcorner hpr new_net_flag file_cnt \
      new_file chan_cnt sta_file rt k x num_sta sta_cnt s_serial datalog \
      sign stay dbout s_rate fr_files resp_file_cnt new_net rt

   #puts "this is the time before reset: $time"
   incr time -1
   if {$sign != "OFF"} {
    set i $stay
  }

  set num_sta($time) $i
       #set i 1
  mkdir $new_net
  mkdir $new_net/stapars
  #puts "chan_cnt($i) is $chan_cnt($i)"
  #puts "what's the time? $time"
  set time 1
  set fix $i
  set num_sta($time) $i
  set new_time($time) $rt($time)
  #puts "num_sta is in write_field is: $num_sta($time)"

  #puts "file_cnt in write_field is : $file_cnt"
  #puts "what's i? $i"
  #puts "what's j? $j"
  set path $new_net/stapars
  #puts "in proc write_field the site is $site($i,$t($time))"
  #puts "rt($time) is :$rt($time)"
       set new_file($time) $path/$new_time($time)
  #puts "newfile($time) is at $new_file($time)"
  while {$time <= $file_cnt} {
     set num_sta($time) $fix
     set new_file($time) $path/$new_time($time)
     #puts "the 2nd newfile($time) is at $new_file($time)"
     set sta_file($time) [open $new_file($time) a+]
     puts $sta_file($time) "Site ? _SID __Latitude _Longitude \
        _Elevation N-offset E-offset V-offset"
     set i 1
     loc_block_out  $site($i,$t($time)) $who($i,$t($time)) \
           $sid($i,$t($time)) $lat($i,$t($time)) $long($i,$t($time))\
           $elev($i,$t($time)) $noff($i,$t($time))\
           $eoff($i,$t($time)) $voff($i,$t($time)) $sta_file($time)
     set i 1
     set j 1
     puts $sta_file($time) "Site Comp Type Serial Theta Phi ____Freq \
               ____Damp ____Coil RsponseFile Sens Depth"
     sensor_out  $site($i,$t($time)) $comp($i,$j,$t($time)) \
           $type($i,$j,$t($time)) $s_serial($i,$j,$t($time)) \
           $theta($i,$j,$t($time)) $phi($i,$j,$t($time)) \
           $freq($i,$j,$t($time)) $damp($i,$j,$t($time)) \
           $coil($i,$j,$t($time)) $rfile($i,$j,$t($time)) \
           $sens($i,$j,$t($time)) $depth($i,$j,$t($time))
     set i 1
     set x 1
     set j 1
     puts $sta_file($time) "_SID Serial DataLogger SampleRate \
               FilterResponseFiles_____________________________"
     logger_out $sta_file($time) $sid($i,$t($time)) \
               $serial($i,$j,$t($time)) $datalog($i,$j,$t($time))\
               $s_rate($i,$j,$t($time)) $fr_files($i,$x,$t($time))
     set i 1
     set k 1
     puts $sta_file($time) "_SID Comp dbPreAmp __dbGain Corner \
               Roll Counts/V HighCorner _HPR"
     amp_out  $sid($i,$t($time)) $comp($i,$k,$t($time)) \
               $preamp($i,$k,$t($time)) $gain($i,$k,$t($time)) \
               $corner($i,$k,$t($time)) $roll($i,$k,$t($time)) \
               $cv($i,$k,$t($time)) $hcorner($i,$k,$t($time)) \
               $hpr($i,$k,$t($time)) $sta_file($time)
     #set n 1
     #if {$id_flag == "net"} {
```

```
#
#     puts $sta_file($time) "Net_Type Net_Label Buf_Size BcdOff \
                TagOff TagSiz WrdSiz Npts Start_time__________"
#     net_out $n_type($n,$t($time)) $n_label($n,$t($time)) \
                $buff_siz($n,$t($time)) $b_off($n,$t($time))\
#               $tag_off($n,$t($time)) $tag_siz($n,$t($time)) \
                $w_siz($n,$t($time)) $npts($n,$t($time)) \
#               $s_time($n,$t($time)) $sta_file($time)
##        }
      close $sta_file($time)
      incr time
   }
}


######################################
#set proper flags and path.
######################################

proc pre_start {} {
   global pattern i j k t time site comp rt type serial theta \
        phi freq damp coil rfile sens depth file_flag Wcnt

   set sec_arg $pattern
   set dir $pattern
   set dir_flag $pattern
   set file_flag 0
   set Wcnt 0
   start
}

###########################################
# just the under the construction message.
###########################################

proc uc {} {
        set w .uc
        if { [winfo exists $w] } {
                wm deiconify $w
                blt::winop raise $w
                return
                }
        toplevel $w
        label $w.program \
                -text "Under Construction" \
                -font -*-helvetica-bold-r-*-*-*-240-*-*-*-*-*-*
        label $w.date \
                -text "" \
                -font -*-helvetica-bold-r-*-*-*-140-*-*-*-*-*-*
        label $w.version \
                -text "Come Back Later!" \
                -font -*-helvetica-bold-r-*-*-*-240-*-*-*-*-*-*

        button $w.dismiss -text Dismiss -command "wm withdraw $w"
        pack $w.program $w.date $w.version $w.dismiss \
                  -anchor center -fill x
}

###############################################
# Initialize the stapar table window and its variables
###############################################

proc table_init {} {
    global many_boxes counter db pattern new_time time new new_net_flag rt
    set w [uniqueW ""]
    toplevel $w

    set new_time($time) $rt($time)
    set table "table"
    set time 1
    set nrecords $counter
    switch $new_net_flag {

        0 { set name $new}
        1 { set name $new_time($time)}
    }
    #set name $new
    set basename [file tail $name]

    wm title $w $name
    wm iconname $w $basename
    #wm iconbitmap $w tableicon
    set mb [menu_setup $w]

    set n 10
    set nchars 0
    while { $n < $nrecords } {
        set nchars [expr $nchars+1]
        set n [expr $n*10]
     }


    global Entry 1
    label $w.name -text DETAIL
    label $w.record -textvariable Record($w) -width $nchars

    button $w.ok -text "ok" -state disabled -command "uc"
    button $w.abort -text "X" -state disabled -command "uc"
    entry $w.entry -relief sunken -textvariable Entry($w)
    bind $w.entry <Return> "uc"


    scrollbar $w.row \
        -orient vertical \
        -command  \
           {multi_scroll $many_boxes } \
        -width 10

    bind $w.row <Any-Button> "+set Selection($w) {}"

    global tkPriv
    bind $w.row <Any-ButtonRelease> {
        incr tkPriv(buttons) -1
        incr t_flag
#        table_fill [winfo parent %W]
        }

    scrollbar $w.column \
        -orient horizontal \
        -command "show_col $w" \
        -width 10

    bind $w.column <Any-ButtonRelease> {
        incr tkPriv(buttons) -1
        #incr col_flag
        fix_columns [winfo parent %W]
        }
    label $w.maxrecord -textvariable Nrecords($w) -width $nchars
    button $w.dismiss    -text Dismiss   -command "destroy $w"
    blt::table $w \
        $mb        2,0 -columnspan 100 -anchor w -fill x \
        $w.ok      3,0                 -anchor w -fill x \
        $w.abort   3,1                 -anchor w -fill x \
        $w.entry   3,2 -columnspan 100 -anchor w -fill x \
        $w.record 10,0                 -anchor w -fill x \
        $w.row    11,0 -rowspan 10  -fill y \
        $w.maxrecord 25,0 -anchor w -fill x \
        $w.column 25,1 -columnspan 100 -fill x \
        $w.dismiss 30,0 -columnspan 100 -fill x

        #$mb        2,0 -columnspan 100 -anchor w -fill x \
    global MinHeight MinWidth
    if { ! [info exists MinHeight] } {
        set MinHeight [expr [winfo reqheight $mb] \
                + [winfo reqheight $w.ok] \
                + 2 * [winfo reqheight $w.row] \
                + [winfo reqheight $w.row] \
                + [winfo reqheight $w.dismiss] ]

        set MinWidth  [winfo reqwidth $w.row]
        }

#    table_init_global $db $database $table $w
    bind $w <Configure> {
        # puts stdout "%W Configure: height=%h width=%w send_event=%E Ok=$Ok2resize(%W)"
        # if { [info exists Wwidth(%W)] } {
        #     puts stdout "\t Previous height=$Wheight(%W) Previous width=$Wwidth(%W)"
        #     }

        if { [info exists Ok2resize(%W)] && $Ok2resize(%W) } {
            if { %E && ( $Wheight(%W) != %h || $Wwidth(%W) != %w) } {
                set Ok2resize(%W) 0
#                compute_rows %W %h
                set Wheight(%W) %h
                set Wwidth(%W) %w
                set Resize_width(%W) %w
                set Maxflag(%W) 0
#                fix_rows %W $Nrows(%W)
#                fix_columns %W
    #            table_fill %W
                set Wheight(%W) %h
                set Wwidth(%W) %w
                }
          } else {
            if { %E } {
                set Ok2resize(%W) 1
                set Wheight(%W) %h
                set Wwidth(%W) %w
                }
            }
        }

    global Maxwidth Maxheight
    #3wm maxsize $w $Maxwidth(.) $Maxheight(.)

return $w
}

##################################
# the new multi_scrolling deal.
```

```
################################

proc multi_scroll { scroll_list args } {
   set len [llength $scroll_list]

   for {set i 0} {$i < $len} {incr i} {
        set temp_list [lindex $scroll_list $i]

#       puts "Command: $temp_list yview $args"

        eval $temp_list yview $args
   }
}



#######################################################
# Initialize the number of rows displayed by the table
#######################################################

proc compute_rows { w {size 0} } {
    global Db Nrows Nrows_default Wheight CharHeight

    set db $Db($w)
    if { $size > 0 } {
        set nrows [expr int(($size-110)/$CharHeight)]
    } else {
        set nrows $Nrows_default
        }
    #set Nrows($w) [max 1 [min $nrows [dbquery $db dbRECORD_COUNT]]]
    }
proc fix_rows { w n } {
    global Name Nchars Nrows
    set Nrows($w) $n
    foreach c [winfo children $w] {
        if { [regexp $w.lb $c] } {
            set nchars $Nchars($Name($c))
            $c configure -width ${nchars} -height $n
            }
        }
    }


#######################################################
# Copy the currently selected field to the entry
#######################################################

proc set_entry { lb name } {
    global Mode \
            Editok \
            Parent \
            w \
            selection \
            Entry \
            Expression \
            Selection \
            Old \
            Selectionlb \
            Selectioncol \
            Selectionindex

    set w $Parent($name)
puts "parent is $w"
```

```
#    save_edit $w 1
    set selection [$lb curselection]
puts "the selection is $selection"

    set Selection($w) $selection
    # $w.mb.delete configure -state normal
    if { [llength $selection] == 1 } {
        set Entry($w) [$lb get $selection]
        set Old($w) $Entry($w)
        #if { $Editok($w) && ! [is_expression $Expression($name)]} {
        #    set Mode($w) 1
            set Selectioncol($w) $name
            set Selectionlb($w) $lb
            set Selectionindex($w) $selection

#             $w.ok configure -state normal
#             $w.abort configure -state normal
#             }
        }
    }

#######################################################
# Initialize the table window's global variables
#
# This is part of Dan's code, just trying to see
# how it works with the previous procedure
#######################################################

proc table_init_global {database table w spe} {
    global Db Request Mapped

    set Db spe

    set Mapped ""

    global Record
    set Record($w) 0

    global Nrecords
    set Nrecords($w) [dbquery $db dbRECORD_COUNT]

    global Maxwidth Maxheight
    set Maxwidth($w) $Maxwidth(.)
    set Maxheight($w) $Maxheight(.)

    global Mode Selection
    set Mode($w) 0
    set Selection($w) ""

    global Ok2resize
    set Ok2resize($w) 0
}

#####################################################
# WRITE TABLE
# not a very good name for this procedure, but
# it makes all the directories for css30 data.
#####################################################

proc writetable {} {
   global j chan_cnt file_cnt numfield num_sta rt cnt t i new_net time line \
                sens comp dbout main_fid aff_fid sid rfile s_rate datalog x \
                site dl_flag chanid cali_fid instr_fid sid net_id sens_id \
                s_rate db_fid next first match stage_fid chan_fid lat long lon elev \
                 cnt cv coil noff eoff depth phi theta s_serial site_fid \
                 stage_info ext sensors stage_dir ch channel rsponse_dir sitename
   if {$time != 1} {
      incr time -1
   }
   set i 1
   set j 1
   set first 0
   set next 1
   set ch 0
   set file_cnt [expr ($file_cnt - 1)]
   set channel($i,$j,$t($time)) $ch
   set chanid $channel($i,$j,$t($time))
   set match ""
   set sensors ""
   mkdir $new_net/css30
   mkdir $new_net/css30/field
   mkdir $new_net/css30/field/response
   mkdir $new_net/css30/field/response/stage
   set rsponse_dir " $new_net/css30/field/response"
   set stage_dir " $new_net/css30/field/response/stage"
   set main_fid [open $new_net/stapars/$rt($time) r]
   set id_flag 0
   set j $chan_cnt($i)
      while {[gets $main_fid line] >= 0} {
#puts "line is in writetables: $line"
        set numfield [llength $line]
        set line_indx1 [lindex $line 0]
        set line_indx2 [lindex $line 2]
        if {($line_indx1 == "Site") || ($line_indx1 == "_SID") || \
           ($line_indx1 == "Net_Type")} {
                set id_flag 0
        }

        switch $id_flag {

                loc     {  loc_block_in line i t($time)
                           site_in $sitename($i) $site($i,$t($time)) $lat($i,$t($time)) \
                                $long($i,$t($time)) $elev($i,$t($time)) \
                                $noff($i,$t($time)) $eoff($i,$t($time))
                        }
                sen     {  sensor_in line i j t($time)
                           #chan_in line i j t($time)
                        }
                d_log   {  logger_in line i j x t($time)
                           puts "s_rate($i,$j,$rt($time) $s_rate($i,$j,$t($time))"
                           aff_in $new_net $site($i,$t($time))
                           in_net
                        }
                amp     {  amp_in line i k t($time)

                        }

                default {if {($line_indx1 != "Site") || ($line_indx1 != "_SID") \
                        || ($line_indx1 != "Net_Type")} {
                        if {($numfield == 9) && ($line_indx2 == "_SID")} {
                                        set i 0
                                        set cnt 0
                                        set num_sta($time) $cnt
                                        set id_flag loc
                                } elseif  {($numfield == 12)} {
                                        #set sta_cnt($i,$t($time)) $strg_cnt
                                        set id_flag sen
                                        #set cnt 0
                                        set i 0
                                } elseif  {($numfield == 5)} {
                                        set chan_cnt($i) $j
                                        set id_flag d_log
                                        set i 0
                                        set x 0
                                } elseif  {($numfield == 9) && ($line_indx2 == "dbPreAmp")
} {
                                        set id_flag amp
                                        set resp_file_cnt($i,$t($time)) $dl_flag
                                        set i 0
                                } else {
                                        set id_flag 0
                                }
                        }
                }
        }
      }
#puts "what is num_sta in writetable: $num_sta($time)"

to_table
}

proc to_table {} {
   global dbout i j t time site sens depth phi theta num_sta chan_cnt \
                s_rate rfile datalog sid  file_cnt phi theta depth \
                sitename

   set time 1
puts "time is $time and file_cnt is $file_cnt"

        chan_out $sens($i,$j,$t($time)) $depth($i,$j,$t($time)) \
           $phi($i,$j,$t($time)) $theta($i,$j,$t($time)) $sitename($i)
}

##############################################
# looking at the first line in hf/response dir
# to find information about a certain filter
# or sensor.
##############################################

proc firstline {$line} {
          global Pf control one cnt line stapar t check num_sta theo num alias \
                stage_cnt gtype paz catfile paz target dish stage_info cat_file seismo cat
                          set field_cnt [expr [llength $line] -8]
                          scan $line "%s %s %s %s %s " \
                                  theo num alias paz\
                                  seismo
        set num $cnt
        #set num $stage_cnt
        #set cat_file [open $cat a+]
        set maz [lindex $line 3]

        if {($control == 1} {
                #puts "line in firstline is is: $line"
           puts $cat_file [format "%s %s %s %s %s" $theo $num $alias \
                $paz $seismo]
        }
incr cnt
     }

##########################################################################
# Creating CSS 3.0 output tables.
##########################################################################
```

```
# AFFILIATION TABLE
###################
proc aff_in {$new_net $site($i,$t($time))} {
   global sta net new_net site i t time

   set sta [format "%-6s" $site($i,$t($time))]
   set net [format "%-8s" $new_net]

}

proc aff_out {$new_net $site($i,$t($time)) } {
   global sta net rt new_net site i t time dbout


   if {[catch {dbaddv $dbout affiliation net $net sta $sta \
                } or_add_err]} {
        puts "affiliatiion_out:         $or_add_err"
   }
}

######################
# CALIBRATION TABLE
######################
proc cali_in { $sens($i,$j,$t($time)) $rt($time) \
        $comp($i,$j,$t($time)) $cv($i,$j,$t($time)) $coil($i,$j,$t($time))} {
   global stime sta stream endtime sens site chan rt time t i comp calper \
         cv chanid newrange calib fc units coil

   incr j
   incr chanid
   set sta [format "%-6s" $site($i,$t($time))]
   set calper 0
   set xcalib [expr [expr (1/$cv($i,$j,$t($time))] /$coil($i,$j,$t($time))]
   #set calib [expr ($xcalib * 10e-7)]
   set fc -1.000000
   set units -
   set chan [format "%-8s" $sens($i,$j,$t($time))]
   set stream [format "%8d" $comp($i,$j,$t($time))]
   set stime [format "%17.5lf" $newrange]
   set endtime 9999999999.99900

}

proc cali_out { $sens($i,$j,$t($time)) $rt($time) \
        $comp($i,$j,$t($time))} {
   global i j stime endtime stream chan rt sta sens comp calper calib units fc dbout

   if {[catch {dbaddv $dbout calibration sta $sta chan $chan \
        time $stime endtime $endtime stream $stream calib $calib \
        calper $calper fc $fc units $units} cali_add_err]} {
        puts "calibration_out:          $cali_add_err"
        }
}

#######################
# INSTRUMENT TABLE
#######################
proc instr_in {} {
   global x sid rfile i j t time inid insname instype band digital s_rate ncalib ncalper d
ir \
        chan_cnt rtype band first next match ck_rfile type dfile rsptype dbout samprate da
talog

   puts "chan_cnt($i) is :$chan_cnt($i)"
```

```
    set j $chan_cnt($i)
    set q [expr ($j + 1)]
    set p [expr ($i + 1)]
    set cal [expr ($time - 1)]
    set ck_rfile 1
    #set inid [format "%8d" $sid($i,$t($time))]
    set samprate [format "%11.7lf" $s_rate($i,$j,$t($time))]
    set insname "$rfile($i,$j,$t($time)):$datalog($i,$j,$t($time))"
    set instype [format "%-6s" $rfile($i,$j,$t($time))]
    set digital d
    set ncalib 0
    set ncalper 0
    set dir [file tail /hf/hifreq/response]
    set dfile -
    #set rtype $type($i,$j,$t($time))
}

##################################
# the instrument table
#################################

proc instr_out {} {

    global inid insname instype band digital s_rate ncalib ncalper dir \
         first next match ck_rfile dfile rsptype dbout samprate

    set length [llength $match]

    if {[llength $match] > 1} {
         set one [lindex $match $first]
         set two [lindex $match $next]

         set ck_rfile [string compare $one $two]
       set first [expr ($first + 1)]
       set next [expr ($next + 1)]
    }

    if {$ck_rfile != 0} {
       if {[catch {dbaddv $dbout instrument inid $inid \
            insname $insname instype $instype \
            band $band digital $digital \
            samprate $samprate ncalib $ncalib \
            ncalper $ncalper dir $dir \
            dfile $dfile rsptype $rsptype} instr_add_err]} {
            puts "instrument out:       $instr_add_err"
         }

    }

}

######################
# NETWORK TABLE
######################
proc in_net {} {
    global nettype new_net netname auth commid

    set new_net $new_net
    set netname $netname
    set nettype lo
    set auth "UCSD"
    set commid -1

}
```

```
proc out_net {} {
   global dbout netname nettype new_net auth commid

   if {[catch {dbaddv $dbout network net $new_net netname $netname \
                nettype $nettype auth $auth commid $commid} \
                net_add_err]} {
                puts "network out:      $net_add_err"
        }

}

####################
# SENSOR TABLE
####################

proc sens_in {} {
   global i j time t sta chan chanid endtime inid chanid jdate jday calratio calper \
        yrday newclock tshift instant site sens sid newrange

   set chanid [format "%8d" $chanid]
   set ep [epoch "$yrday$jday"]
   set jdate [yearday "$ep"]
   set calratio 1.000000
   set tshift 0.00
   set instant y
   set sta [format "%-6s" $site($i,$t($time))]
   set chan [format "%-8s" $sens($i,$j,$t($time))]
   #set inid [format "%8d" $sid($i,$t($time))]
   set stime [format "%17.5lf" $newrange]

}

proc sens_out {} {
   global sta chan endtime inid chanid jdate jday calratio calper \
        sid newrange i j time t yrday tshift instant stime dbout
  #set chanid [format "%8d" $chanid]
   #set ep [epoch "$yrday$jday"]
   #set jdate [yearday "$ep"]
   set calratio 1.000000
   set tshift 0.00
   set instant y
   #set sta [format "%-6s" $site($i,$t($time))]
   #set chan [format "%-8s" $sens($i,$j,$t($time))]
   #set inid [format "%8d" $sid($i,$t($time))]
   set stime [format "%17.5lf" $newrange]

   if {[catch {dbaddv $dbout sensor sta $sta chan $chan time $stime \
                endtime $endtime inid $inid chanid $chanid jdate $jdate \
                calratio $calratio calper $calper tshift $tshift \
                instant $instant} sens_add_err]} {
                puts "sensor_out:       $sens_add_err"
        }
}

###################
# SITE TABLE
###################

proc site_in {$sitename($i,$t($time)) $site($i,$t($time)) $lat($i,$t($time)) $long($i,$t($
time)) \
                $elev($i,$t($time)) $noff($i,$t($time)) $eoff($i,$t($time))} {
   global site lat lati long lon elev el newrange ondate offdate staname refsta dnorth dea
st \
	stacode jday yrday newclock i time t statype noff eoff rfsta sitename

   set sta [format "%-6s" $site($i,$t($time))]
   set lati [format "%9.4lf" $lat($i,$t($time))]
   set lon [format "%9.4lf" $long($i,$t($time))]
   set el [format "%9.4lf" $elev($i,$t($time))]
   set ep [epoch "$yrday$jday"]
   set ondate [yearday "$ep"]
   puts "ondate is :$ondate"
   set offdate -1
   if {$stacode == 1} {
        set sitename($i) -
   }
   set staname $sitename($i)
   set statype ss
#   set refsta $rfsta($i,$t($time))
   set dnorth [format "%9.4lf" $noff($i,$t($time))]
   set deast [format "%9.4lf" $eoff($i,$t($time))]

}

proc site_out {} {
   global lat lati lon elev el ondate \
         offdate staname statype refsta dnorth \
        deast dbout
   global site lat lati long lon elev el newrange \
         next rf_sta ondate offdate staname refsta dnorth deast \
        jday yrday newclock i time t statype noff eoff rfsta sitename
#
   set next [expr ($i +1)]

puts "site($i,$t($time)) is $site($i,$t($time))"

#   set sta [format "%-6s" $site($i,$t($time))]
   set sta MLK
   if {$stacode == 1} {
        set sitename($i) -
   }
   set lati [format "%9.4lf" $lat($i,$t($time))]
   set lon [format "%9.4lf" $long($i,$t($time))]
   set el [format "%9.4lf" $elev($i,$t($time))]
   set ep [epoch "$yrday$jday"]
   set ondate [yearday "$ep"]
   puts "ondate is :$ondate"
   set offdate -1
   set staname $sitename($i)
   set statype ss
   set refsta $rf_sta
#   set refsta $rfsta($i,$t($time))
   set dnorth [format "%9.4lf" $noff($i,$t($time))]
   set deast [format "%9.4lf" $eoff($i,$t($time))]

   if {[catch {dbaddv $dbout site sta $sta ondate $ondate offdate $offdate \
                lat $lati lon $lon elev $el staname $staname statype \
                $statype refsta $refsta dnorth $dnorth deast $deast} site_add_err]} {
                puts "site_out:         $site_add_err"
        }
}

###################
# SITECHAN TABLE
###################

proc chan_out {$sens($i,$j,$t($time)) $depth($i,$j,$t($time)) \
        $phi($i,$j,$t($time)) $theta($i,$j,$t($time)) $sitename($i,$t($time))} {
    global  site i chan_cnt num_sta sens j t time dbout sta chan ondate  \
        channel thing depth phi theta chanid ctype edepth hang vang descrip
    global stime sta stream endtime sens site chan rt time t i comp calper \
         minus rtype type cv chanid newrange calib fc units coil eoff noff
    global lat long elev id_cnt endtime jdate jday ep calratio calper \
         offdate identity yrday newclock tshift instant sid newrange who
    global x sid rfile insname instype band digital s_rate \
        dir first next match ck_rfile dfile rsptype dbout samprate datalog
    global when x_flag dbout netname nettype new_net auth commid staname
    global rf_sta site_name stage_info item sta net new_net site sitename rfsta
    global newrange stage_info stageid iunits ounits gtype decifac samples dir dfile
    global dbout sens site s_serial datalog s_rate stage_loop Pf i j t time rfile
    global coil cv serial stid ssident gnom iunts ounts gcalib gtp izero dfac \
         stay then id stage_time stage_info ext rsponse_var smprt leadfac dr dfe
    global file_cnt ok sign mark tableyrday tabletyme tableminus

puts "the num_sta($time) in chan_out is: $num_sta($time)"

    set change [array names sid]
    set id 1
    set find($id) -
    puts "rfile($i,$j,$t($time))_datalog($i,$j,$rt($time)).1"
       set item "$rfile($i,$j,$t($time)):$datalog($i,$j,$rt($time)):$s_rate($i,$j,$rt($time
))"
    if {$sign != "OFF"} {
       set i $stay
    }

  while {$time <=  $file_cnt} {

    for {set i 1} {$i <= $num_sta($time)} {incr i} {
    set chan_ck $chan_cnt($i)
    set comp_ck $chan_cnt($mark)
    #set chan_add [expr ($chan_cnt($i) + $chan_ck)]
    if {($i > 1 || $mark > 1)} {
    #set chan_add [expr ($chan_cnt($i) + $chan_ck)]
    set comp_add [expr ($chan_cnt($i) + $comp_ck)]
    set comp_incr [expr ($comp_add - $chan_cnt($mark))]
#puts "chan_add is $chan_add"
puts "comp_incr is $comp_incr"
#puts "comp_add is $comp_add"
    } else {
         set comp_incr [expr ($chan_cnt($i) - $chan_cnt($mark))]
    }

puts "You big DUMMY!"
    global aff_fid cali_fid instr_fid sens_fid net_fid site_fid \
               chan_fid stage_fid

    set aff_fid [open $new_net/css30/field/$new_net.affiliation a+]
    set cali_fid [open $new_net/css30/field/$new_net.calibration a+]
    set instr_fid [open $new_net/css30/field/$new_net.instrument a+]
    set sens_fid [open $new_net/css30/field/$new_net.sensor a+]
    set net_fid [open $new_net/css30/field/$new_net.network a+]
    set site_fid [open $new_net/css30/field/$new_net.site a+]
    set chan_fid [open $new_net/css30/field/$new_net.sitechan a+]
    set stage_fid [open $new_net/css30/field/$new_net.stage a+]
    set dbout [dbopen $new_net/css30/field/$new_net r+]
    puts "who($i,$t($time)) in chan_out is $who($i,$t($time))"

write2tables
    incr id_cnt
    }
```

```
incr time
   }
}

proc check_insname {} {
global track find insname_match name_match insname_loop insname inid i j t time rfile data
log
   #set check [lsearch $instr_match $stage_list]
   set compare [lsearch $name_match $insname]
   if {$compare == -1} {
      lappend name_match $insname
      incr insname_loop
      set inid [llength $name_match]
puts "inid in check is: $inid"
      set find($insname) $inid
      #set count 1
   } else {
        set inid $find($insname)
   }

}

########################################################
# this is the actual procedure the writes to all tables
# but instrument and stage.
########################################################

proc write2tables {} {
   global  site i chan_cnt num_sta sens j t time dbout sta chan ondate  \
        channel thing depth phi theta chanid ctype edepth hang vang descrip
   global stime sta stream endtime sens site chan rt time t i comp calper \
         rtype type cv chanid newrange calib fc units coil eoff noff
   global lat long elev id_cnt endtime jdate jday ep calratio then calper \
        offdate identity yrday newclock tshift instant sid newrange who
   global when x sid rfile insname instype band digital s_rate \
        dir first next match ck_rfile dfile rsptype dbout samprate datalog
   global ncalib ncalper dbout netname nettype new_net auth commid staname
   global rf_sta site_name stage_info item sta net new_net site sitename rfsta
   global newrange stage_info stageid iunits ounits gtype decifac samples dir dfile
   global dbout sens site s_serial datalog s_rate stage_loop Pf i j t time rfile
   global coil cv serial stid ssident gnom iunts ounts gcalib gtp izero dfac \
         dir ondate id stage_time stage_info ext rsponse_var smprt leadfac dr dfe
   global main_fid sens_fid then2 track inid tableminus find ok mark

   global aff_fid cali_fid instr_fid sens_fid net_fid site_fid \
                minus chan_fid stage_fid stacode tabletime


   set stable 1
   set track $chan_cnt($i)
   for {set j $track} {$j >= 1} {incr j -1} {
   set net [format "%-8s" $new_net]
   set new_net $new_net
   set netname $netname
   set nettype lo
   set auth "UCSD"
   if {$stacode == 1} {
      set when($i) $tabletime($time)
      set then($i) $minus($time)
      set s_rate($i,$j,$t($time)) $s_rate($i,$stable,$t($time))
      set datalog($i,$j,$t($time)) $datalog($i,$stable,$t($time))
      set serial($i,$j,$t($time)) $serial($i,$stable,$t($time))
      set ok($i,$j,$time) $ok($i,$stable,$time)
   }
```

```
#puts "ok($i,$j,$time) is $ok($i,$j,$time) in write2tables"

   if {$ok($i,$j,$time) == 0} {
              set stage_time [string trimleft [string range $when($i) 0 4]]
              #set modify [string trimleft [string range $then($i) 0 4]]
              set o_date [string trimleft [string range $when($i) 0 4]]
     #puts "second modify is: $modify"
#puts "we now want to put an offdate "
#puts "then is: $then($i)"
#puts "when is: $when($i)"
              #set offdate [yearday "$then($i)"]
              #set temp $offdate
              set ondate $when($i)
              set stime $when($i)
              #set endtime $then($i)
#puts "offdate is: $offdate"
#puts "ondate is: $ondate"
   } else {

######################################################################
# if there is a change in a stations information before building the
# the database tables, the endtime and offdate need to be added here.
######################################################################

#puts "in the else!!!!"

        set odate [yearday "$then2($i)"]

        set st_cnt $chan_cnt($mark)
#puts "what is dbRECORD_COUNT: $comp_incr"

        set counter 0
           if {[catch {dbputv $dbout site $counter offdate $odate} \
                site_add_err]} {
              puts "edit_out:           $site_add_err"
           }

        while {$counter < $st_cnt} {
           if {[catch {dbputv $dbout sensor $counter endtime $then2($i)} \
                sens_add_err]} {
              puts "edit_out:           $sens_add_err"
           }
           if {[catch {dbputv $dbout stage $counter endtime $then2($i)} \
                stage_add_err]} {
              puts "edit_out:           $stage_add_err"
           }
           if {[catch {dbputv $dbout calibration $counter endtime $then2($i)} \
                cali_add_err]} {
              puts "edit_out:           $cali_add_err"
           }
           if {[catch {dbputv $dbout sitechan $counter offdate $odate} \
                sitechan_add_err]} {
              puts "edit_out:           $sitechan_add_err"
           }
           incr counter
        }

         site_out

   }

#######################################################
# setting field values and writing to database tables.
# for now.
#######################################################

    set commid -1
    set lati [format "%9.4lf" $lat($i,$t($time))]
    set lon [format "%9.4lf" $long($i,$t($time))]
    set el [format "%9.4lf" $elev($i,$t($time))]
    set dnorth [format "%9.4lf" $noff($i,$t($time))]
    set deast [format "%9.4lf" $eoff($i,$t($time))]
    catch {set samprate [format "%11.7lf" $s_rate($i,$j,$t($time))]}
    catch {set insname "$rfile($i,$j,$t($time)):$datalog($i,$j,$t($time))"}
    #set samprate [format "%11.7lf" $s_rate($i,$j,$rt($time))]
    #set insname "$rfile($i,$j,$t($time)):$datalog($i,$j,$rt($time))"

    if {$stacode == 1} {
         set sitename($i) -
    }
    set staname $sitename($i)
    set thing($i,$j,$t($time)) [format "%-6s" $site($i,$t($time))]
    set sta $thing($i,$j,$t($time))
    set statype ss
    set refsta $rf_sta
    set chan [format "%-8s" $sens($i,$j,$t($time))]
    set chid($i,$j,$t($time)) $id
    set cid $chid($i,$j,$t($time))
    set ctype n
    set edepth [format "%9.4lf" $depth($i,$j,$t($time))]
    set hang   [format "%6.1lf" $phi($i,$j,$t($time))]
    set vang   [format "%6.1lf" $theta($i,$j,$t($time))]
    set descrip -
    set calper 0
    set jdate [yearday "$ep"]
    set xcalib [expr (1/$cv($i,$j,$t($time))/$coil($i,$j,$t($time)))]
    set calib [expr ($xcalib * 10e06)]
    set calib [format "%16.6lf" $calib]
    set fc -1.000000
    set units "nm/sec/cnt"
    set stream [format "%8d" $comp($i,$j,$t($time))]
    set jdate [yearday "$ep"]
    set calratio 1.000000
    set tshift 0.00
    set instant y
    #set inid [format "%8d" $sid($i,$t($time))]
    set ck_rfile 1
    set instype [format "%-6s" $rfile($i,$j,$t($time))]
    set rtype($i,$j,$t($time)) $type($i,$j,$t($time))
    set rsptype [string trimleft [string range $rtype($i,$j,$t($time)) 0 0]]
               set o_date [string trimleft [string range $when($i) 0 4]]
    #set band -
    set digital d
    set ncalib 0
    set ncalper 0
    set dir [file tail /hf/hifreq/response]
    global rsponse_var
    compute_id
    compute_stage

puts "this is the time before we write to tables: $t($time) $when($i)"
puts "what about the offtime: $then($i)"

    if {[catch {dbaddv $dbout sensor sta $sta chan $chan time $stime \
                endtime $endtime inid $inid chanid $cid jdate $jdate \
                calratio $calratio calper $calper tshift $tshift \
                instant $instant} sens_add_err]} {
```

```
                puts "sensor_out:       $sens_add_err"
        }
    if {[catch {dbaddv $dbout sitechan sta $sta chan $chan ondate $ondate chanid $cid \
                offdate $offdate ctype $ctype edepth $edepth hang $hang \
                vang $vang descrip $descrip} chan_add_err]} {
                puts "sitechan_out:     $chan_add_err"
    }
    if {[catch {dbaddv $dbout calibration sta $sta chan $chan \
        time $stime endtime $endtime stream $stream calib $calib \
        calper $calper fc $fc units $units} cali_add_err]} {
        puts "calibration_out:          $cali_add_err"
        }
    if {[catch {dbaddv $dbout site sta $sta ondate $ondate offdate $offdate \
                lat $lati lon $lon elev $el staname $staname statype \
                $statype refsta $refsta dnorth $dnorth deast $deast} site_add_err]} {
                puts "site_out:         $site_add_err"
        }
    if {[catch {dbaddv $dbout network net $new_net netname $netname \
                nettype $nettype auth $auth commid $commid} \
                net_add_err]} {
                puts "network out:      $net_add_err"
        }
    if {[catch {dbaddv $dbout affiliation net $net sta $sta \
                } or_add_err]} {
        puts "affiliatiion_out:         $or_add_err"
    }

    incr id
    }
    close $aff_fid
    close $instr_fid
    close $site_fid
    close $sens_fid
    close $net_fid
    close $chan_fid
    close $stage_fid
    close $cali_fid
    dbclose $dbout
}

#################################################################
# computing the insname field and writing the instrument table
#################################################################

proc compute_id {} {
    global match rtype rfile s_rate datalog i j t time dbout inid insname \
         rsponse_var item instype band digital samprate ncalib ncalper dfile rsptype
    global insname_loop find inid rt dir info_flag ext stage_info list_cnt \
          freq instr_match sensors stacode

    set stage_info \
          "$rfile($i,$j,$t($time)):$datalog($i,$j,$rt($time)):$s_rate($i,$j,$rt($time))"
    set stage_list\
          "$rfile($i,$j,$t($time))_$datalog($i,$j,$rt($time))"
    set check [lsearch $instr_match $stage_list]
    set compare [lsearch $match $stage_info]
    set frequency $freq($i,$j,$t($time))
    set band [calc_band $samprate $frequency]

    if {$compare == -1} {
       lappend match $stage_info
       set check [lsearch $instr_match $stage_list]
       set count 1
       incr insname_loop
```

```
      set inid [llength $match]
      #set find($stage_list) $inid
#puts "instr_match is $instr_match and stage_list is $stage_list"
         if {$check != -1} {
            lappend instr_match $stage_list
      set count 0
      foreach num $instr_match {
         if {$stage_list == $num} {
            incr count
         }
      }
            set dfile $stage_list.$count
            set find($stage_info) $inid
#puts "inid in check is: $inid"
         } else {
            lappend instr_match $stage_list
            set dfile $stage_list.$count
            set find($stage_info) $inid
#puts "inid in check is: $inid"
         }
         #set dir [file tail /hf/hifreq/response]
#puts "the appended match is $match"
puts "this is the instrument table!!!!!!!"
         if {[catch {dbaddv $dbout instrument inid $inid \
            insname $insname instype $instype \
            band $band digital $digital \
            samprate $samprate ncalib $ncalib \
            ncalper $ncalper dir $dir \
            dfile $dfile rsptype $rsptype} instr_add_err]} {
            puts "instrument out:       $instr_add_err"
           }
#exit
   } else {
        set inid $find($stage_info)
   }
}

##################
#destroys slaves.
##################

proc leave {w} {

        destroy $w
}

####################################
#This procedure  deletes a field
#from the table .
####################################

proc del_col {name i} {
        global Parent Column Column_label Expression Request one
        set l $Column($name)
        set b $Column_label($name)
puts "what's b: $b  $l"
        blt::table forget $b
        blt::table forget $l
}

proc show_record { w } {
        global Record Nrows Nrecords counter firstrow
        set Nrecords($w) $counter
        set Nrows($w) 20
        set firstrow [max  0 $firstrow ]
        set firstrow [min $firstrow [expr $Nrecords($w)-1]]
        set Record($w) $firstrow
        $w.row set $Nrecords($w) $Nrows($w) $firstrow [expr $firstrow+$Nrows($w)-1]
puts "Nrecords($w) is actually: $Nrecords($w)"

        puts "first is:$firstrow"
        puts "and: [expr $firstrow+$Nrows($w)-1]"
}


#
# Set the horizontal scrollbar to indicate the fields displayed
#
proc show_col {w firstrow} {
    global Mapped Request First_col request
    global Column_starts Maxwidth Min_window_width

        global Record Crows Crecords counter
        set count [llength $request]
        #set Nrecords($w) $counter
        set Crecords($w) $count
        set Crows($w) 20
        set firstrow [max  0 $firstrow ]
        set firstrow [min $firstrow [expr $Crecords($w)-1]]
        set Record($w) $firstrow
        set First_col($w) 7
        puts "first is:$firstrow"
        puts "and: [expr $firstrow+$Crows($w)-1]"
        $w.column set $Crecords($w) $Crows($w) $firstrow [expr $firstrow+$Crows($w)-1]
    #$w.column set $total $shown $left $right
    }

#
# Set the horizontal scrollbar to indicate the fields displayed
#
proc table_fill { w } {
       global Record Nrows Nrecords firstrow i j file_cnt
        global Column Column_label Column_number counter

        set firstrow $Record($w)

       set Nrows($w) 20
       set nrecords $counter
       set Nrecords($w) $nrecords
       wm geometry $w ""

        set Record($w) $firstrow
       set record $Record($w)
        puts "the rec is $Record($w)"
        puts "and: [expr $Record($w)+$Nrows($w)-1]"
#       del $w $1
       $w.row set $Nrecords($w) $Nrows($w) $Record($w) [expr $Record($w)+$Nrows($w)-1]
}
#
# Return a unique window name
#
proc uniqueW { w } {
    global Wcnt
    set Wcnt [expr $Wcnt+1]
    return $w.w$Wcnt
}

proc uniqueN {} {
        global Wcnt
        set Wcnt [expr $Wcnt+1]
        return $Wcnt
    }
    proc uniqueF { } {
        global Fcnt
        set Fcnt [expr $Fcnt+1]
        return .f$Fcnt
    }

    ############################################################
    #calls the start procedure in order to reset the variables.
    ############################################################
    proc put_back {} {
       global new_net rt new_time files new file_cnt time pattern\
            new_net_flag file_flag site i t

       incr i -1
       set time 1
       set file_flag 0
    puts "file count now is: $file_cnt"
            set new_net_flag 1
            set pattern $new_net
            #start
        if {$file_cnt > 1} {
            set pattern $new_net/stapars
            all
        } else {
            pre_start
        }

            writetable
            #to_table
    }

    proc clear {} {
        global rt time new_time new_time_flag new_sta_flag file_cnt \
            chan_cnt i j

        #set new_time($time) $rt($time)
        set file_cnt 0
        set new_net_flag 0
        set new_sta_flag 0
        set new_time_flag 0
        set time 0
        set j 0
        set i 0

    }

    ############################################################
    # creates the interactive text window to allow the user to
    # create and edit stapars, and build field database tables.
    ############################################################

    proc create_stapars {} {
        global new_comp time count field_list comp i j t new_net \
                chan_cnt new_time new_file site serial_ck \
                new_time_flag new_sta_flag count twin win2 \
                rf_sta comp_cnt yr jday hr min sec rt v q site_cnt

        set f [uniqueF]
        toplevel $f
        set z 2
    set v 1
    set q [expr ($v - 1)]
      set width 15
      set name "Create $new_net Stapars"
      set new_time_flag 1
      set new_sta_flag 1

      wm title $f $name
      wm geometry $f +0+0

  frame $f.m1 -relief raised
  frame $f.m2
     set twin $f.m1.text
     set win2 $f.m2

  text $f.m1.text -relief sunken -bd 2 -yscrollcommand "$f.m1.scroll set"
  scrollbar $f.m1.scroll -command "$f.m1.text yview"
  button $f.m2.new_time     -text "New Time" \
                          -state disabled \
                          -command "newfield $f"
  button $f.m2.new_sta              -text "New Station" \
                          -state disabled \
                          -command "newsite $f"
  button $f.m2.edit_sta              -text "Edit Station" \
                          -state disabled \
                          -command "EditStation $f"
  button $f.m2.new_comp     -text "New Component" \
                          -state disabled \
                          -command "newcomp $f"
  button $f.m2.db           -text "Build Database" \
                          -state disabled \
                          -command { \
                                write_field; \
                                put_back; \
                                }
  button $f.m2.off          -text "Exit" \
                                -command "destroy $f"
  button $f.m2.done         -text "Done" \
                         -state disabled \
                          -command "send $f "

  pack $f.m1 -side top
  pack $f.m2 -side bottom
  pack $f.m2.off $f.m2.new_time $f.m2.new_sta $f.m2.edit_sta  \
           $f.m2.new_comp $f.m2.db $f.m2.done  \
          -side left

  #pack $f.m2.off $f.m2.new_time $f.m2.new_sta $f.m2.edit_sta  \
  #        $f.m2.new_comp $f.m2.db $f.m2.done  \
  #       -side left -ipadx 2m -ipady 1m

  pack $f.m1.scroll -side right -fill y
  pack $f.m1.text -side left

  puts "the value of f is $f"
  enter $f
  }

  proc send { f } {
     global w win2 db_flag time num_sta
     global i comp_flag comp_cnt question1 j chan_cnt
       if {$comp_flag > 0} {
          #incr i
          set comp_flag 0
```

```
        set comp_cnt 1
        set question1 1
#       incr num_sta($time)
        incr j -1
        set chan_cnt($i) $j
#       set db_flag 1
        enter $f
        }
}

#######################################################
# this is where all the text for the create stapars
# comes from.
#######################################################

proc enter { f } {
   global time twin win2 count i j file_cnt time_flag \
        sta_flag comp_flag db_flag question1 num_sta \
        comp_cnt v q time_cnt site_cnt edit1

  $f.m1.text tag add x $v.0 $v.end
  if {$time_flag == 1} {
  set seiko "enter time $time_cnt\n"
  $f.m1.text insert end $seiko
  $f.m1.text tag add x $v.0 $v.end
  $f.m1.text configure -font -*-bookman-*-*-*-*-*-*-*-*-*-*-*-*
  $f.m1.text configure -foreground black
  $f.m2.new_time configure -state active
     $f.m2.db configure -state disabled
     $f.m2.edit_sta configure -state disabled
     set time_flag 0
     #set site_cnt 1
  incr v
  }
  if {$sta_flag == 1} {
     set station "enter station $site_cnt\n"
     $f.m1.text insert end $station
  $f.m1.text tag add z $q.0 $q.end
  $f.m1.text tag add a $v.0 $v.end
  $f.m1.text tag configure a -foreground black
  $f.m1.text tag configure x -foreground black
     $f.m1.text configure -font -*-bookman-*-*-*-*-*-*-*-*-*-*-*-*
  $f.m1.text configure -foreground black
     $f.m2.db configure -state disabled
     $f.m2.new_time configure -state disabled
     $f.m2.edit_sta configure -state disabled
     $f.m2.new_sta configure -state active
     set sta_flag 0
  incr v
  }
  if {$comp_flag == 1} {
     set channel "enter component $comp_cnt\n"
     $f.m1.text insert end $channel
  $f.m1.text tag add c $q.0 $q.end
  $f.m1.text tag add b $v.0 $v.end
  set yo [$f.m1.text tag ranges x]
  $f.m1.text tag configure b -foreground black
  $f.m1.text tag configure a -foreground black
     $f.m1.text configure -font -*-bookman-*-*-*-*-*-*-*-*-*-*-*-*
  $f.m1.text configure -foreground black
     $f.m2.new_time configure -state disabled
     $f.m2.new_sta configure -state disabled
     $f.m2.edit_sta configure -state disabled
     $f.m2.db configure -state disabled
```

```
      $f.m2.new_comp configure -state active
      set db_flag 0
   incr v
   }
   if {$comp_flag > 1} {
   $f.m1.text tag configure b -foreground black
      set channel "enter component $comp_cnt or done\n"
      $f.m1.text insert end $channel
   $f.m1.text tag add d $q.0 $q.end
   $f.m1.text tag add e $v.0 $v.end
   $f.m1.text tag configure e -foreground black
   $f.m1.text tag configure b -foreground black
      $f.m1.text configure -font -*-bookman-*-*-*-*-*-*-*-*-*-*-*-*
   $f.m1.text configure -foreground black
      $f.m2.db configure -state disabled
      $f.m2.edit_sta configure -state disabled
      $f.m2.done configure -state active
      $f.m2.new_comp configure -state active
      set db_flag 0
   incr v
   }
   if {$db_flag == 1} {
      set css "ready to build database\n"
      $f.m1.text insert end $css
   $f.m1.text tag add f $q.0 $q.end
   $f.m1.text tag add g $v.0 $v.end
   $f.m1.text tag configure g -foreground black
   $f.m1.text tag configure e -foreground black
      $f.m1.text configure -font -*-bookman-*-*-*-*-*-*-*-*-*-*-*-*
   $f.m1.text configure -foreground black
      $f.m2.new_comp configure -state disabled
      $f.m2.edit_sta configure -state disabled
      $f.m2.done configure -state disabled
      $f.m2.db configure -state active
      incr i -1
      set num_sta($time) $i
      set db_flag 0
   incr v
   }
   if {$question1 == 1} {
      set quest1 "enter station $site_cnt or time $time_cnt or build database?\n"
      $f.m1.text insert end $quest1
   $f.m1.text tag add z $q.0 $q.end
   $f.m1.text tag add x $v.0 $v.end
   $f.m1.text tag configure z -foreground black
   $f.m1.text tag configure g -foreground black
      $f.m1.text configure -font -*-bookman-*-*-*-*-*-*-*-*-*-*-*-*
   $f.m1.text configure -foreground black
      $f.m2.new_comp configure -state disabled
      $f.m2.done configure -state disabled
      $f.m2.new_sta configure -state active
      $f.m2.edit_sta configure -state active
      $f.m2.new_time configure -state active
      $f.m2.db configure -state active
      set question1 0
      set db_flag 0
      set time_flag 0
   incr v
   }
   start_time $f
}

###########################################
# creates a window for entering the time.
```

```
##########################################

proc newfield { f } {
   global  i file_cnt sta_flag question1 tfields db_flag comp_flag time
   global fieldname p time_cnt site_go
   set y 0
   incr time_cnt
   set cell [uniqueW ""]
   #set cell .c
   toplevel $cell
   set name "TIME"
puts "the value of cell is $cell"

   set sta_flag 1
   set question1 0
    wm title $cell $name
    wm geometry $cell +0-0

   set tfields 0
   set comp_flag 0
   set db_flag 0
   set p 1
   incr time
puts "time in newfiled is : $time"

   set MasterFrame [frame $cell.g ]
   set SubFrame [frame $MasterFrame.f -relief groove]
   set g1 [frame $SubFrame.g1 ]
   set k0 [label $g1.k -text "Start Time:"]
   set k1 [BuildEntry $g1.1 len "Year" 10 15]
   set k2 [BuildEntry $g1.2 len "Jday" 10 15]
   set k3 [BuildEntry $g1.3 len "Hour" 10 15]
   set k4 [BuildEntry $g1.4 len "Min" 10 15]
   set k5 [BuildEntry $g1.5 len "Sec" 10 15]

   pack $g1.k $g1.1 $g1.2 $g1.3 $g1.4 $g1.5 \
        -padx 2 \
        -anchor e \
        -side top \
        -in $g1

   pack $g1 -side left -anchor n -fill x -in $SubFrame
   pack $SubFrame -side top -in $MasterFrame

   set cancel [button $MasterFrame.but \
        -text "Cancel" \
        -command "destroy $cell"]

   set next [button $MasterFrame.nxt \
        -text "Continue" \
        -command "incr file_cnt; first_time $f; destroy $cell"]

   pack $cancel $next -side bottom -anchor w -fill x -in $MasterFrame

   pack $MasterFrame -in $cell

   #####################################
   # setting the focus window
   #####################################

   focus $k1
   bind $k1 <Return> "set Add(year) Add($fieldname(1));set_time;  focus $k2"
   bind $k1 <Tab> "set Add(year) Add($fieldname(1));set_time;  focus $k2"
   bind $k2 <Return> "set Add(jday) Add($fieldname(2));set_time; focus $k3"
   bind $k2 <Tab> "set Add(jday) Add($fieldname(2));set_time; focus $k3"
   bind $k3 <Return> "set Add(hour) Add($fieldname(3));set_time; focus $k4"
   bind $k3 <Tab> "set Add(hour) Add($fieldname(3));set_time; focus $k4"
   bind $k4 <Return> "set Add(minute) Add($fieldname(4));set_time; focus $k5"
   bind $k4 <Tab> "set Add(minute) Add($fieldname(4));set_time; focus $k5"
   bind $k5 <Return> "set Add(second) Add($fieldname(5));set_time; \
                      incr file_cnt; first_time $f; destroy $cell"
   bind $k5 <Tab> "set Add(second) Add($fieldname(5));set_time; \
                      incr file_cnt; first_time $f; destroy $cell"

   global Add


}

##############################################################
# For the first time thru creating new stations and channels,
# the respective entry windows will come up automatically.
##############################################################

proc start_time { f } {
   global i sta_flag site_go
   if {$site_go == 0} {
   if {$i == 0} {
      #set site_go 1
        newfield $f
        #set sta_flag 0
   }
      set site_go 1
   }
}

proc first_time { f } {
   global  site_go time netscape
      if { $time == 1 } {
           enter $f
           newsite $f
      } else {enter $f}

   set site_go 1
}

proc first_time_comp { f } {
   global  time netscape
   if { $time == 1 } {
        enter $f
        newcomp $f
   } else {enter $f}
}

proc netval {} {
   global new_net netname Add fieldname
        set new_net $Add($fieldname(1))
        set netname $Add($fieldname(2))
}

#######################################################
# this is the site entry window.
#######################################################

proc newsite { f } {
   global new_comp time count field_list site t i \
               sitename num_sta time chan_cnt new_time serial_ck \
               sign stay sta_cnt sid new_net new_sta_flag comp_flag \
                        site_cnt code c new_time j tfields this next rt cell

    global fieldname p Add MasterFrame Site Who DSerial DataLogger\
         SampleRate db_flag N_offset E_offset V_offset Elevation
    global sign ck_numsta mark x_flag Longitude Latitude
    set cell [uniqueW ""]
    toplevel $cell
    set name "STATION INFO"
    wm title $cell $name
    wm geometry $cell +0-0
    incr i
    set x_flag($i) -
    set j 1
    set code 1
    set serial_ck 0
    if {$sign == "OFF"} {
       set num_sta($time) $i
    } else {
          set i $mark
          set num_sta($time) $stay }

       set ck_numsta($time) $num_sta($time)

 #puts "num_sta($time) in newsite is : $num_sta($time)"

    set t($time) $rt($time)
    #set sta_cnt($i,$t($time)) $i
    set new_file($time) $rt($time)
    set name "Station"
    set sid($i,$t($time)) $i
 #puts "sid($i,$t($time)) is $sid($i,$t($time))"
    incr site_cnt
    #set i $site_cnt
    set comp_flag 1
    set p 1
    set c 0
    set z 3
    set MasterFrame [frame $cell.g ]
    set SubFrame [frame $MasterFrame.f -relief groove]
    set g1 [frame $SubFrame.g1 ]
    set k($c) [label $g1.k -text "Station Information:"]
            pack $g1.k  \
                 -padx 2 \
                 -anchor w \
                 -side top \
                 -in $g1

       set fieldlist " {$Site} StationName {$Latitude}  \
         {$Longitude} {$Elevation}  {$N_offset} {$E_offset} \
         {$V_offset} ReferenceStation"

         foreach field_val $fieldlist {
            incr c
            if {$field_val == "StationName"} {
                 set k($c) [BuildEntry $g1.$c len $field_val 50 25]
            } else { set k($c) [BuildEntry $g1.$c len $field_val 10 25]
            }

 #puts "k is $k($c)"
            pack $g1.$c \
                 -padx 2 \
                 -anchor w \
                 -side top \
                    -in $g1
          }

     pack $g1 -side left -anchor n -fill x -in $SubFrame
     pack $SubFrame -side top -in $MasterFrame

     set cancel [button $MasterFrame.but \
          -text "Cancel" \
          -command "incr i -1; destroy $cell"]

     set next [button $MasterFrame.nxt \
          -text "Continue" \
          -command "incr count; enter $f; destroy $cell"]

     pack $cancel $next -side bottom -anchor w -fill x -in $MasterFrame

     pack $MasterFrame -in $cell

     incr c -1

     #############################################
     # for default values and previously entered
     # values.
     #############################################

     if {$sign == "OFF"} {
        if {$i <= 1} {
           glob_vals
        }
     fixedvals
     } else { change_vals }


     global Add
     focus $k(1)
     bind $k(1) <Return> "select; \
                               focus $k(2)"
     bind $k(1) <Tab> "select; \
                              focus $k(2)"
     bind $k(2) <Return> "select; focus $k(3)"
     bind $k(2) <Tab> "select; focus $k(3)"
     bind $k(3) <Return> "select; focus $k(4)"
     bind $k(3) <Tab> "select; focus $k(4)"
     bind $k(4) <Return> "select; focus $k(5)"
     bind $k(4) <Tab> "select; focus $k(5)"
     bind $k(5) <Return> "select;  focus $k(6)"
     bind $k(5) <Tab> "select;  focus $k(6)"
     bind $k(6) <Return> "select; focus $k(7)"
     bind $k(6) <Tab> "select; focus $k(7)"
     bind $k(7) <Return> "select; focus $k(8)"
     bind $k(7) <Tab> "select; focus $k(8)"
     bind $k(8) <Return> "select; focus $k(9)"
     bind $k(8) <Tab> "select; focus $k(9)"
     bind $k(9) <Return> "select;  \
                              incr count; \
                              ripple_back; \
                                  ripple_up; first_time_comp $f; destroy $cell"
     bind $k(9) <Tab> "select;  \
                              incr count; \
                              ripple_back; \
                                  ripple_up; first_time_comp $f; destroy $cell"
  }

  proc glob_vals {} {
   global comp_cnt fieldname p Add code

   switch $code {

      1 { set Add($fieldname(1)) -
          set Add($fieldname(2)) -
          set Add($fieldname(3)) -999.0000
          set Add($fieldname(4)) -999.0000
          set Add($fieldname(5)) -999.0000
          set Add($fieldname(6)) 0
          set Add($fieldname(7)) 0
          set Add($fieldname(8)) 0
          set Add($fieldname(9)) -
        }
      2 { set Add($fieldname(1)) $comp_cnt
          set Add($fieldname(2)) 0
          set Add($fieldname(3)) -
          set Add($fieldname(4)) 0
          set Add($fieldname(5)) 0
          set Add($fieldname(6)) 0.0
          set Add($fieldname(7)) 0.0
          set Add($fieldname(8)) 0.0
          set Add($fieldname(9)) 0.0
          set Add($fieldname(10)) 0
          set Add($fieldname(11)) -
          set Add($fieldname(12)) -1.0000000
        }
   }
}

proc change_vals {} {
   global comp_cnt fieldname p Add code
   global time t site who sid lat long elev noff eoff voff i Add comp sens \
        type serial theta phi freq damp coil rfile depth j new_time new_comp \
        preamp gain corner roll cv hcorner hpr new_net_flag file_cnt \
        new_file s_serial x serial_ck datalog s_rate fr_files resp_file_cnt\
        position code yr jday hr min sec file_cnt rt fieldname rfsta sitename \
        rbtime mark then when x_flag num_sta cnt identity

   switch $code {

      1 { set Add($fieldname(1)) $site($mark,$t($rbtime))
          set Add($fieldname(2)) $sitename($mark)
          set Add($fieldname(3)) $lat($mark,$t($rbtime))
          set Add($fieldname(4)) $long($mark,$t($rbtime))
          set Add($fieldname(5)) $elev($mark,$t($rbtime))
          set Add($fieldname(6)) $noff($mark,$t($rbtime))
          set Add($fieldname(7)) $eoff($mark,$t($rbtime))
          set Add($fieldname(8)) $voff($mark,$t($rbtime))
          set Add($fieldname(9)) -
        }
      2 { set Add($fieldname(1)) $comp_cnt
          set Add($fieldname(2)) $s_serial($mark,$j,$t($rbtime))
          set Add($fieldname(3)) $rfile($mark,$j,$t($rbtime))
          set Add($fieldname(4)) -
          set Add($fieldname(5)) $depth($mark,$j,$t($rbtime))
          set Add($fieldname(6)) $preamp($mark,$j,$t($rbtime))
          set Add($fieldname(7)) $gain($mark,$j,$t($rbtime))
          set Add($fieldname(8)) $hcorner($mark,$j,$t($rbtime))
          set Add($fieldname(9)) $hpr($mark,$j,$t($rbtime))
          set Add($fieldname(10)) $serial($mark,$j,$t($rbtime))
          set Add($fieldname(11)) $datalog($mark,$j,$t($rbtime))
          set Add($fieldname(12)) $s_rate($mark,$j,$t($rbtime))
        }
      }
   }
   proc fixedvals {} {

      global comp_cnt fieldname p Add code

      switch $code {

         1 { set Add($fieldname(1)) $Add($fieldname(1))
             set Add($fieldname(2)) $Add($fieldname(2))
             set Add($fieldname(3)) $Add($fieldname(3))
             set Add($fieldname(4)) $Add($fieldname(4))
             set Add($fieldname(5)) $Add($fieldname(5))
             set Add($fieldname(6)) $Add($fieldname(6))
             set Add($fieldname(7)) $Add($fieldname(7))
             set Add($fieldname(8)) $Add($fieldname(8))
             set Add($fieldname(9)) $Add($fieldname(9))
           }
         2 { set Add($fieldname(1)) $comp_cnt
             set Add($fieldname(2)) $Add($fieldname(2))
             set Add($fieldname(3)) $Add($fieldname(3))
             set Add($fieldname(4)) $Add($fieldname(4))
             set Add($fieldname(5)) $Add($fieldname(5))
             set Add($fieldname(6)) $Add($fieldname(6))
             set Add($fieldname(7)) $Add($fieldname(7))
             set Add($fieldname(8)) $Add($fieldname(8))
             set Add($fieldname(9)) $Add($fieldname(9))
             set Add($fieldname(10)) $Add($fieldname(10))
             set Add($fieldname(11)) $Add($fieldname(11))
             set Add($fieldname(12)) $Add($fieldname(12))
           }
     }


   }

   #################################################
   # this is the channel information entry window.
   #################################################

   proc newcomp { f } {
      global code new_comp time count field_list comp i j t new_net \
      chan_cnt sign new_time new_file site serial_ck \
      comp_cnt new_time_flag new_sta_flag question1 comp_flag \
                     c new_time j tfields this next rt db_flag


      global fieldname p Add Comp SSerial Coil dbPreAmp dbGain Corner\
           Depth Orientation Model Roll CountsV Highcorner HPR Freq Damp
      global net_path DSerial DataLogger SampleRate Pf

      set cell [uniqueW ""]
      toplevel $cell
      set z 3
      set width 15
      incr serial_ck


      set c 0
      set code 2
      set tfields 0
      set x 1
      set y 1
      set next ""
```

```
set db_flag 0
incr comp_flag
set p 1
set name "Component"
wm title $cell $name
wm geometry $cell +0-0

set chan_cnt($i) $j

pfgetarr logger_list %${Pf}#LoggersList
pfgetarr sensor_list %${Pf}#ResponseFile

set MasterFrame [frame $cell.g ]
set SubFrame [frame $MasterFrame.f -relief groove]
set g1 [frame $SubFrame.g1 ]
set k($c) [label $g1.k -text "Component Information:"]
        pack $g1.k  \
             -padx 2 \
             -anchor w \
             -side top \
             -in $g1

menubutton $g1.m -text "     datalogger type" -menu $g1.m.b
menubutton $g1.m1 -text "    sensor type" -menu $g1.m1.b
pack $g1.m $g1.m1 -anchor w -side top -in $g1

menu $g1.m.b
menu $g1.m1.b
$g1.m.b add cascade -label "UCSD" -menu $g1.m.b.open
$g1.m1.b add cascade -label "UCSD" -menu $g1.m1.b.open
menu $g1.m.b.open
menu $g1.m1.b.open
for_array_keys list logger_list {
     $g1.m.b.open add command -label $list -command "uc"
}
$g1.m.b.open add command -label "Other?" -command "uc"

for_array_keys list2 sensor_list {
     $g1.m1.b.open add command -label $list2 -command "uc"
}
$g1.m1.b.open add command -label "Other?" -command "uc"



   set fieldlist " {$Comp} {$SSerial} {$Model} Orientation  \
                    {$Depth} {$dbPreAmp} {$dbGain} \
                    {$Highcorner} {$HPR} {$DSerial} \
                    {$DataLogger} {$SampleRate} "

     foreach field_val $fieldlist {
        incr c
        set k($c) [BuildEntry $g1.$c len $field_val 10 25]

        pack $g1.$c \
             -padx 2 \
             -anchor e \
             -side top \
             -in $g1
     }

pack $g1 -side left -anchor n -fill x -in $SubFrame
pack $SubFrame -side top -in $MasterFrame

set cancel [button $MasterFrame.but \
     -text "Cancel" \
     -command "destroy $cell"]

set next [button $MasterFrame.nxt \
     -text "Next Component" \
     -command "incr j; \
               incr comp_cnt; \
               set chan_cnt($i) $j; \
               enter $f;  \
               destroy $cell"]

pack $cancel $next -side bottom -anchor w -fill x -in $MasterFrame

pack $MasterFrame -in $cell

incr c -1
          if {$comp_cnt <= 1} {
             glob_vals
           }

fixedvals
global Add
focus $k(1)
bind $k(1) <Return> "select; focus $k(2)"
bind $k(1) <Tab> "select: focus $k(2)"
bind $k(2) <Return> "select; focus $k(3)"
bind $k(2) <Tab> "select; focus $k(3)"
bind $k(3) <Return> "select; compute_rfile; focus $k(4)"
bind $k(3) <Tab> "select; compute_rfile; focus $k(4)"
bind $k(4) <Return> "select; compute_position; focus $k(5)"
bind $k(4) <Tab> "select; compute_position; focus $k(5)"
bind $k(5) <Return> "select; focus $k(6)"
bind $k(5) <Tab> "select; focus $k(6)"
bind $k(6) <Return> "select; focus $k(7)"
bind $k(6) <Tab> "select; focus $k(7)"
bind $k(7) <Return> "select; focus $k(8)"
bind $k(7) <Tab> "select; focus $k(8)"
bind $k(8) <Return> "select; focus $k(9)"
bind $k(8) <Tab> "select; focus $k(9)"
bind $k(9) <Return> "select; focus $k(10)"
bind $k(9) <Tab> "select; focus $k(10)"
bind $k(10) <Return> "select; focus $k(11)"
bind $k(10) <Tab> "select; focus $k(11)"
bind $k(11) <Return> "select; check_logger; focus $k(12)"
bind $k(11) <Tab> "select; check_logger; focus $k(12)"
bind $k(12) <Return> "select; \
                      get_fr_files; \
                      compute_channel_codes; \
                      ripple_back; \
                      ripple_up; \
                      CheckTableValues; \
                      incr comp_cnt; \
                      incr j; \
                      set chan_cnt($i) $j; \
                      enter $f; \
                      destroy $cell"
bind $k(12) <Tab> "select; \
                      get_fr_files; \
                      compute_channel_codes; \
                      ripple_back; \
                      ripple_up; \
                      CheckTableValues; \
                      incr comp_cnt; \
                      incr j; \
                                set chan_cnt($i) $j; \
                                enter $f; \
                                destroy $cell"
}

#################################
# generic table entry window
#################################

proc BuildEntry {cell item text size length} {

   global Add show fieldname p
   set num [uniqueN]
   set fieldname($p) $text
   set frame [frame $cell -relief flat -width 50 ]
        set k1 [menubutton $cell.m_$num\
                -menu $cell.m_$num.m  \
                -text $text  \
                -width $length]

        set e [entry $cell.$item \
                -relief sunken \
                -exportselection yes \
                -textvariable Add($fieldname($p)) \
                -width $size]

    menu $cell.m_$num.m
    $cell.m_$num.m add cascade \
        -label "" \
        -background bisque3 \
        -label $text\
        -menu $cell.m_$num.m.detail
        menu $cell.m_$num.m.detail -foreground blue
        detail $cell.m_$num.m.detail

        #pack $k1 -side left -padx 3 -in $frame
        #pack $e -side left -padx 3 -in $frame
        #pack $e  -anchor e -in $frame

        blt::table $cell \
           $cell.m_$num $p,1 -anchor w -fill x \
           $cell.$item  $p,4 -anchor w -fill x

        incr num
        incr p
        return $cell.$item
}

proc detail { m } {
   set line "information will be in place at a later time."
   $m add command -label $line

}


 proc addfield { f } {
    global Add time tail x y t time sp_fields one Possible parse \
        new_time new_comp field_val field_list fieldname count \
        new_net_flag file_cnt num_sta chan_cnt i j new_file \
        fieldname comp_flag sta_flag yr jday hr min sec new_net\
        tfields this rt question1 next x_flag

    set z 3
    set w [uniqueW ""]
    toplevel $w
    set y 0
    set r 1
    set c 0
    set i -
    set x_flag($i) -
    set width 5
    set next($c) ""
    set sta_flag 1
    set question1 0
    set tfields 0
    set comp_flag 0
    set db_flag 0
    set name $new_net.time
    wm title $w $name
    wm iconname $w $name
    incr time
    set t_format "year jday hour min sec"

    puts "what's the file count? $file_cnt"

    foreach date $t_format {

    set fieldname $date
    incr c
puts "c is : $c"

    menubutton $w.m_$fieldname -menu $w.m_$fieldname.m -text $fieldname
    entry $w.e_$fieldname -relief sunken \
                -width $width \
                -exportselection yes \
                -textvariable Add($fieldname)

    bind $w.e_$fieldname <Tab> " incr tfields; set_time $fieldname; \
                nextfield $w $c $fieldname"
    bind $w.e_$fieldname <Return> "incr tfields; set_time $fieldname; \
                nextfield $w $c $fieldname"

        set newclock [str2epoch "$yr$jday$hr$min.$sec"]
    set length [string length $newclock]
    set last [expr $length - 5]
    set newrange [string trimright [string range $newclock 0 $last]]
    set new S
    set rt($time) [concat $newrange$new]
    puts "the conversion is $rt($time)"

    blt::table $w \
        $w.m_$fieldname                         $r,0 -anchor w -fill x \
        $w.e_$fieldname                         $r,3 -anchor w
    incr r
    }
    set this $c
    button $w.cancel     -text Cancel \
                                -command " destroy $w"
    button $w.next      -text Continue \
                                -command "set i 1; incr file_cnt; enter $f; lower $w"

    blt::table $w \
        $w.cancel                       8,0 -anchor w -fill x \
        $w.next                         8,3 -anchor w -fill x
}

proc setsite { } {
   global fieldname Add site sid who lat long elev
```

```
    set st $Add($fieldname(1))
    set sd $Add($fieldname(2))
    set lt $Add($fieldname(3))
    set lg $Add($fieldname(4))
    set ev $Add($fieldname(5))
    set no $Add($fieldname(6))
    set eo $Add($fieldname(7))
    set vo $Add($fieldname(8))
    set ds $Add($fieldname(9))
    set dl $Add($fieldname(10))
    set sr $Add($fieldname(11))
    set fr $Add($fieldname(12))
puts "what's st: $st"
puts "what's st: $sd"
puts "what's st: $lt"
puts "what's st: $lg"
puts "what's st: $ev"
puts "what's st: $no"

puts "what's st: $eo"
puts "what's st: $vo"
puts "what's st: $ds"
puts "what's st: $dl"
puts "what's st: $sr"
puts "what's st: $fr"
}

#######################################################################
# converts human time into perspective database time formats that
# are needed.
#######################################################################

proc set_time { } {
   global newrange fieldname m show yr jday rt hr min sec Add i time \
        tableminus sign minus epoch_clk yrday newclock new_time clock
   global tabletime tabletyme tableyrday

      set yr " $Add($fieldname(1))"
      set jday " $Add($fieldname(2))"
      set hr " $Add($fieldname(3))"
      set min " $Add($fieldname(4))"
      set sec " $Add($fieldname(5))"

    set sign "OFF"
    set yrday $yr
    set jday [string trimright [string range $jday 1 3]]
    set hr [string trimright [string range $hr 1 2]]
    set min [string trimright [string range $min 1 2]]
    set sec [string trimright [string range $sec 1 2]]
    set len [string length $yr]
    if { $len == 2} {
        set yr [string trimright [string range $yr 1 2]]
    } else {
        set last [expr $len - 2]
        set yr [string trimright [string range $yr $last end]]
    }

    set newclock [str2epoch "$yr$jday$hr$min.$sec"]
    set tabletime($time) [str2epoch "$yr:$jday:$hr:$min:$sec"]
    set minus($time) [expr ($tabletime($time) - .1)]
    set tableminus($time) [yearday "$minus($time)"]
    set minustime [strtime "$minus($time)"]
    set tableyrday($time) [yearday "$tabletime($time)"]
```

```
        set tabletyme($time) [strtime "$tabletime($time)"]
    puts "the tabletime is now $tabletime($time)"
    puts "minus is now $minus($time)"
    puts "the newclock is :$newclock"
    puts "the newclock is :$tableyrday($time)"
        set length [string length $newclock]
        set last [expr $length - 5]
        set newrange [string trimright [string range $newclock 0 $last]]
        set new S
        set rt($time) [concat $newrange$new]
        set clock $rt($time)
            set new_time($time) $rt($time)
        set ep [epoch "$yrday$jday"]
        set year [yearday "$ep"]
        set date [strtime "$ep"]
        set stdate [strdate "$ep"]
        set epoch_clk($time) "($year) $date"
        set epoch_clk($time) "($tableyrday($time)) $tabletyme($time)"
        set epch($time) "($tableminus($time)) $minustime"
        puts "the date is:  $epoch_clk($time)"
        puts "with the minus the time is : $epch($time)"

    }

    proc new_network {f } {
        global Add time tail x y t time sp_fields one Possible parse \
            new_time new_comp width field_val field_list fieldname count \
            j new_net_flag file_cnt chan_cnt num_sta new_file sta_cnt file_flag\
            new_time_flag sta_flag new_comp_flag i clear time_flag comp_flag \
            new_net db_flag question1 count comp_cnt
       global  i file_cnt sta_flag question1 tfields db_flag comp_flag time
       global fieldname p time_cnt
        set z 3
        set w [uniqueW ""]
        toplevel $w
        set i 0
        set file_flag 1
        set x 0
        set y 0
        #set p 1
        set width 15
        set count 1
        set next ""
        set sta_flag 0
        set comp_flag 0
        set comp_cnt 1
        set question1 0
        set time_flag 1
        set db_flag 0
        set chan_cnt($i) 0
        set name "NEW NETWORK"
        set text1 "Network code:"
        set text2 "Network name:"
        wm title $w $name
        wm geometry $w +200+300
        set fieldname(1) $text1
        set fieldname(2) $text2


        label $w.label -text $text1
        entry $w.net -relief sunken \
                    -width 10 \
                    -exportselection yes \
                    -textvariable Add($fieldname(1))
    label $w.label2 -text $text2
    entry $w.net2 -relief sunken \
            -width 50 \
            -exportselection yes \
            -textvariable Add($fieldname(2))

    button $w.db                -text Build_Database... \
                                -state disabled \
                                -command {mkdir $new_net
                                          write_field
                                          put_back
                                          }

    button $w.enter             -text Enter      \
                                -command "netval; clear; create_stapars; destroy $w"
    button $w.cancel            -text Cancel     \
                                -command "destroy $w"


    if {$new_net_flag == 1} {
        $w.db configure -state active
    }

    blt::table $w \
        $w.label                1,0 -anchor w  \
        $w.label2               2,0 -anchor w -fill x \
        $w.net                  1,2 -anchor w  \
        $w.net2                 2,2 -anchor w -fill x \
        $w.cancel               $z,0 -anchor w -fill x \
        $w.enter                $z,2 -anchor w -fill x

   focus $w.net
   bind $w.net <Return> "netval; focus $w.net2"
   bind $w.net <Tab> "netval; focus $w.net2"
   bind $w.net2 <Return> "netval; \
                 clear; create_stapars; destroy $w"
   bind $w.net2 <Tab> "netval; \
                 clear; create_stapars; destroy $w"
   #glob_vals
puts "the value of w is $w"
   global Add
}

proc lookNlogs {} {
   global result line_indx4 line_indx5 line_indx6 location counter

   set ref [open reflog r+]
   set ref_out [open r_out a+]
   set location 0
   set counter 0
   set result 0

   while {[gets $ref line] >= 0} {

        set answer 0
        set numfield [llength $line]
        set line_indx1 [lindex $line 0]
        set line_indx2 [lindex $line 1]
        set line_indx3 [lindex $line 3]
        set line_indx4 [lindex $line 2]
        set line_indx5 [lindex $line 3]
        set line_indx6 [lindex $line 4]
        if {($line_indx1 == "Station") && \
                ($line_indx2 == "Name")} { \
            puts $ref_out "$line_indx3"
         }
         if {($line_indx2 == "GPS:") &&  \
                ($line_indx4 == "POSITION:")} {
            compute_GPS $line
            set result [expr ($location + $result)]
            set mean [expr ($result / $counter)]
            puts "result is $result"
            puts "the average is $mean"
            puts $ref_out $line
         }

     }

  }

  proc compute_GPS { line } {
     global result line_indx4 line_indx5 line_indx6 location counter

     set const 60
     set line_indx7 [lindex $line 5]
     set first [string index $line_indx5 0]
     set deg [string trimleft [string range $line_indx5 1 2]]
     set min [string trimleft [string range $line_indx5 4 5]]
     set sec [string trimleft [string range $line_indx5 7 end]]
      #set cat [string trimleft [string range $locate 3 end]]
     #puts "the first index is $first"
     #puts "the second index is $deg"
     #puts "the third index is $min"
     #puts "the fourth index is $sec"

     set location [expr ($deg + ($min + $sec/$const)/$const)]
     puts "the position is:  $location"
     incr counter

  }

  proc sta_info {f } {
      global Add time tail x y t time sp_fields one Possible parse \
          new_time new_comp width field_val field_list fieldname count \
          j new_net_flag file_cnt chan_cnt num_sta new_file sta_cnt file_flag\
          new_time_flag sta_flag new_comp_flag i clear time_flag comp_flag \
          new_net db_flag question1 count comp_cnt
     global  i file_cnt sta_flag question1 tfields db_flag comp_flag time
     global fieldname p time_cnt
      set z 3
      set cell [uniqueW ""]
      toplevel $cell
      set i 0
      set x 0
      set y 0
      set name "Station Log"
      set text1 "Station name:"
      set text2 "Datalogger serial #:"
      set text3 "Datalogger type:"
      set text4 "Mode(PASCAL,Broadband,...):"
      set list "$text1 $text2 $text3 $text4"
      wm title $cell $name
      wm geometry $cell +200+300
      set fieldname(1) $text1
      set fieldname(2) $text2

     set p 1
     set c 0
```

```
    set z 3
    set MasterFrame [frame $cell.g ]
    set SubFrame [frame $MasterFrame.f -relief groove]
    set g1 [frame $SubFrame.g1 ]
    set k($c) [label $g1.k -text "Station Information:"]
            pack $g1.k  \
                 -padx 2 \
                 -anchor w \
                 -side top \
                 -in $g1

       set fieldlist " {$text1} {$text2}  \
         {$text3} {$text4}"

         foreach field_val $fieldlist {
            incr c
            if {$field_val == "StationName"} {
                 set k($c) [BuildEntry $g1.$c len $field_val 50 25]
            } else { set k($c) [BuildEntry $g1.$c len $field_val 30 25]
            }

            pack $g1.$c \
                 -padx 2 \
                 -anchor w \
                 -side top \
                 -in $g1
         }

    pack $g1 -side left -anchor n -fill x -in $SubFrame
    pack $SubFrame -side top -in $MasterFrame

    set cancel [button $MasterFrame.but \
         -text "Cancel" \
         -command "incr i -1; destroy $cell"]

    set next [button $MasterFrame.nxt \
         -text "Continue" \
         -command "incr count; enter $f; destroy $cell"]

    pack $cancel $next -side bottom -anchor w -fill x -in $MasterFrame

    pack $MasterFrame -in $cell

    incr c -1
    global Add
}

#####################################################
#####################################################
# edit station window. gives the user the option of
# station the edit.
#####################################################

proc EditStation { f } {
    global site who i j t time edit1
    global  edit1 i file_cnt sta_flag question1 tfields db_flag comp_flag time
    global fieldname p time_cnt select
     set z 3
     set edit1 1
     set w [uniqueW ""]
     toplevel $w
     set name "Edit Station"
     wm title $w $name
    label $w.label -text "Station:"
    entry $w.net -relief sunken \
            -width 10 \
            -exportselection yes \
            -textvariable select
    label $w.label2 -text "Edit Station:"


    button $w.edit              -text Edit      \
                                -width 10 \
                                -command "set edit1 1;\
                                          ChangeSite; \
                                          newsite $f; \
                                          destroy $w "
    button $w.toff              -text "Turn Off"  \
                                -width 10 \
                                -command "TurnOff; \
                                       destroy $w"
    button $w.cancel            -text Cancel    \
                                -width 10 \
                                -command "destroy $w"

    blt::table $w \
        $w.label2               1,0 -anchor w  \
        $w.label                3,2 -anchor w  \
        $w.net                  3,3 -anchor w  \
        $w.toff                 6,0 -anchor w -fill x \
        $w.edit                 6,2 -anchor w -fill x \
        $w.cancel               6,3 -anchor w -fill x

   focus $w.net
   bind $w.net <Return> "TurnOff"
   bind $w.net <Tab> "TurnOff"

}

############################################
# deactivate a station.
############################################
proc TurnOff {} {
   global Ok select
   global time t site who sid lat long elev noff eoff voff i Add comp sens \
        type serial theta phi freq damp coil rfile depth j new_time new_comp \
        preamp gain corner roll cv hcorner hpr new_net_flag file_cnt \
        new_file s_serial x serial_ck datalog s_rate fr_files resp_file_cnt\
        position code yr jday hr min sec file_cnt rt fieldname rfsta sitename \
        mark stay minus edit1 then num_sta cnt chan_cnt offdate identity

   set msg "Are You Sure You Want To Turn Off $select?"
   dialog .t {TURNOFF} $msg {YES} {NO}
   if {$Ok == 0} {
   foreach index [array names site] {
   puts "the index is: $index"
      if {$site($index) == "$select"} {
puts "sid($index) is: $sid($index)"
          set mark $sid($index)
          set identity($mark,$t($time)) "ON"
          set then($mark) $minus($time)
          set who($mark,$t($time)) "N"
puts " who($mark,$t($time)) is now $who($mark,$t($time))"
      }
   }
  }
```

```
}

########################################################################
# brings all the previous values up for a station about to be edited.
########################################################################

proc ChangeSite {} {
   global Ok select
   global time t site who sid lat long elev noff eoff voff i Add comp sens \
        type serial theta phi freq damp coil rfile depth j new_time new_comp \
        preamp gain corner roll cv hcorner hpr new_net_flag file_cnt \
        new_file s_serial x serial_ck datalog s_rate fr_files resp_file_cnt\
        position code yr jday hr min sec file_cnt rt fieldname rfsta sitename \
        rbtime mark sign stay edit1 then num_sta cnt chan_cnt offdate identity

   set msg "Are You Sure You Want To Edit $select?"
   dialog .t {TURNOFF} $msg {YES} {NO}
   if {$Ok == 0} {
   foreach index [array names site] {
   puts "the index is: $index"
      if {$site($index) == "$select"} {
puts "sid($index) is: $sid($index)"
          set mark $sid($index)
          set identity($mark,$t($time)) "ON"
          set sign $identity($mark,$t($time))
          set who($mark,$t($time)) "Y"
puts " who($mark,$t($time)) is now $who($mark,$t($time))"
      }
   }
   set stay $i
   set next [expr ($i + 1)]
   #set i [expr ($mark -1)]
   set rbtime [expr ($time -1)]

       set cnt 1
       set x 1
       set sid($next,$t($time)) $sid($mark,$t($rbtime))
       puts "sid($next,$t($time)) is $sid($next,$t($time))"
       set position($i,$j,$t($rbtime)) -
       puts "the code in ripple_up is: $code"

        set site($next,$t($time)) $site($mark,$t($rbtime))
        #set sitename($i) $Add($fieldname(2))
        #set who($next,$t($time)) $who($mark,$t($rbtime))
        set lat($next,$t($time)) $lat($mark,$t($rbtime))
        set long($next,$t($time)) $long($mark,$t($rbtime))
        set elev($next,$t($time)) $elev($mark,$t($rbtime))
        set noff($next,$t($time)) $noff($mark,$t($rbtime))
        set eoff($next,$t($time)) $eoff($mark,$t($rbtime))
        set voff($next,$t($time)) $voff($mark,$t($rbtime))
        #set rfsta($next,$t($time)) $rfsta($mark,$t($rbtime))
        set resp_file_cnt($,$t($time)) 0
        for {set element $j} {$element <= $chan_cnt($i)} {incr element} {

           set comp($next,$element,$t($time)) $comp($mark,$element,$t($rbtime))
           set type($next,$element,$t($time)) $type($mark,$element,$t($rbtime))
           set s_serial($next,$element,$t($time)) $s_serial($mark,$element,$t($rbtime))
           set rfile($next,$element,$t($time)) $rfile($mark,$element,$t($rbtime))
           #set position($next,$element,$t($time)) $position($mark,$element,$t($rbtime))
           set depth($next,$element,$t($time)) $depth($mark,$element,$t($rbtime))
           set preamp($next,$element,$t($time)) $preamp($mark,$element,$t($rbtime))
           set gain($next,$element,$t($time)) $gain($mark,$element,$t($rbtime))
           set hcorner($next,$element,$t($time)) $hcorner($mark,$element,$t($rbtime))
           set hpr($next,$element,$t($time)) $hpr($mark,$element,$t($rbtime))
```

```
                set theta($next,$element,$t($time)) $theta($mark,$element,$t($rbtime))
                set phi($next,$element,$t($time)) $phi($mark,$element,$t($rbtime))
                set freq($next,$element,$t($time)) $freq($mark,$element,$t($rbtime))
                set cv($next,$element,$t($time)) $cv($mark,$element,$t($rbtime))
                set damp($next,$element,$t($time)) $damp($mark,$element,$t($rbtime))
                set coil($next,$element,$t($time)) $coil($mark,$element,$t($rbtime))
                set sens($next,$element,$t($time)) $sens($mark,$element,$t($rbtime))
                set corner($next,$element,$t($time)) $corner($mark,$element,$t($rbtime))
                set roll($next,$element,$t($time)) $roll($mark,$element,$t($rbtime))
                set serial($next,$element,$t($time)) $serial($mark,$element,$t($rbtime))
                set datalog($next,$element,$t($time)) $datalog($mark,$element,$t($rbtime))
                set s_rate($next,$element,$t($time)) $s_rate($mark,$element,$t($rbtime))
                set fr_files($next,$x,$t($time)) $fr_files($mark,$x,$t($rbtime))
            }
     }
}

######################################
# create new station names.
######################################


proc reset {} {
   global i comp_flag comp_cnt question1 j chan_cnt
        #incr i
        set comp_flag 0
        set comp_cnt 1
        set question1 1
        incr num_sta($time)
        incr j
        set chan_cnt($i) $j
        create_stapars
}
###############################################################
# produces a selection window in order to create new stapars.
###############################################################

#############################################################
# Store new data back into arrays to output into new stapars.
#############################################################

proc select {} {

   global time t site who sid lat long elev noff eoff voff i Add comp sens \
        type serial theta phi freq damp coil rfile depth j new_time new_comp \
        preamp gain corner roll cv hcorner hpr new_net_flag file_cnt \
        new_file s_serial x serial_ck datalog s_rate fr_files resp_file_cnt\
        position code yr jday hr min sec file_cnt rt fieldname rfsta sitename \
        ok tabletime sign then when x_flag num_sta cnt identity
set cnt 1
set x 1
if {$x_flag($i) != "OFF"} {
   set when($i) $tabletime($time)
   #set when($i) $t($time)
   set x_flag($i) "OFF"
}
if {$sign == "OFF"} {
   set who($i,$t($time)) Y
   set then($i) 9999999999.99900
} else { set who($i,$t($time)) "N" }

set position($i,$j,$t($time)) -
set ok($i,$j,$time) 0
puts "position($i,$j,$t($time)) is $position($i,$j,$t($time))"
```

```
set identity($i,$t($time)) "OFF"
puts "identity($i,$t($time)) is :$identity($i,$t($time))"

   switch $code {

      1 {       set site($i,$t($time)) $Add($fieldname(1))
                set sitename($i) $Add($fieldname(2))
                #set who($i,$t($time)) Y
                set identity($i,$t($time)) "OFF"
                set lat($i,$t($time)) [format "%f" $Add($fieldname(3))]
                set long($i,$t($time)) [format "%f" $Add($fieldname(4))]
                set elev($i,$t($time)) [format "%f" $Add($fieldname(5))]
                set noff($i,$t($time)) [format "%f" $Add($fieldname(6))]
                set eoff($i,$t($time)) [format "%f" $Add($fieldname(7))]
                set voff($i,$t($time)) [format "%f" $Add($fieldname(8))]
                set rfsta($i,$t($time)) $Add($fieldname(9))
                set resp_file_cnt($i,$t($time)) 0
        }
      2 {       set comp($i,$j,$t($time)) [format "%d" $Add($fieldname(1))]
                set s_serial($i,$j,$t($time)) [format "%d" $Add($fieldname(2))]
                set rfile($i,$j,$t($time)) $Add($fieldname(3))
                set position($i,$j,$t($time)) [format "%s" $Add($fieldname(4))]
                set depth($i,$j,$t($time)) [format "%d" $Add($fieldname(5))]
                set preamp($i,$j,$t($time)) [format "%f" $Add($fieldname(6))]
                set gain($i,$j,$t($time)) [format "%f" $Add($fieldname(7))]
                set hcorner($i,$j,$t($time)) [format "%f" $Add($fieldname(8))]
                set hpr($i,$j,$t($time)) [format "%f" $Add($fieldname(9))]
                set serial($i,$j,$t($time)) [format "%d" $Add($fieldname(10))]
                set datalog($i,$j,$t($time)) $Add($fieldname(11))
                set s_rate($i,$j,$t($time)) [format "%f" $Add($fieldname(12))]
        }
   }
          #set errmsg "data just enterd is in the wrong format!"


}

proc CheckTableValues {} {
   global time t i j site lat long elev noff eoff voff rfsta comp \
        s_serial rfile position depth preamp gain hcorner hpr serial \
        theta phi edit1 datalog minus s_rate ok then2

   set back [expr ($time - 1)]
   #set j [expr ($j - 1)]
   set ok($i,$j,$time) 0
puts "what's ok($i,$j,$time): $ok($i,$j,$time)"

  if {$edit1 == 1} {
puts "edit is $edit1"
   if {$time > 1} {
      if {[string match $site($i,$t($time)) $site($i,$t($back))] != 1} {
         set ok($i,$j,$time) 1
      } elseif {[string match $lat($i,$t($time)) $lat($i,$t($back))] != 1} {
         set ok($i,$j,$time) 1
      } elseif {[string match $long($i,$t($time)) $long($i,$t($back))] != 1} {
         set ok($i,$j,$time) 1
      } elseif {[string match $elev($i,$t($time)) $elev($i,$t($back))] != 1} {
         set ok($i,$j,$time) 1
      } elseif {[string match $noff($i,$t($time)) $noff($i,$t($back))] != 1} {
         set ok($i,$j,$time) 1
      } elseif {[string match $eoff($i,$t($time)) $eoff($i,$t($back))] != 1} {
         set ok($i,$j,$time) 1
      } elseif {[string match $voff($i,$t($time)) $voff($i,$t($back))] != 1} {
         set ok($i,$j,$time) 1
        } elseif {[string match $s_serial($i,$j,$t($time)) $s_serial($i,$j,$t($back))] != 1}
    {
           set ok($i,$j,$time) 1
        } elseif {[string match $comp($i,$j,$t($time)) $comp($i,$j,$t($back))] != 1} {
           set ok($i,$j,$time) 1
        } elseif {[string match $theta($i,$j,$t($time)) $theta($i,$j,$t($back))] != 1} {
           set ok($i,$j,$time) 1
        } elseif {[string match $phi($i,$j,$t($time)) $phi($i,$j,$t($back))] != 1} {
           set ok($i,$j,$time) 1
        } elseif {[string match $rfile($i,$j,$t($time)) $rfile($i,$j,$t($back))] != 1} {
           set ok($i,$j,$time) 1
        } elseif {[string match $depth($i,$j,$t($time)) $depth($i,$j,$t($back))] != 1} {
           set ok($i,$j,$time) 1
        } elseif {[string match $preamp($i,$j,$t($time)) $preamp($i,$j,$t($back))] != 1} {
           set ok($i,$j,$time) 1
        } elseif {[string match $gain($i,$j,$t($time)) $gain($i,$j,$t($back))] != 1} {
           set ok($i,$j,$time) 1
        } elseif {[string match $hcorner($i,$j,$t($time)) $hcorner($i,$j,$t($back))] != 1} {
           set ok($i,$j,$time) 1
        } elseif {[string match $hpr($i,$j,$t($time)) $hpr($i,$j,$t($back))] != 1} {
           set ok($i,$j,$time) 1
        } elseif {[string match $serial($i,$j,$t($time)) $serial($i,$j,$t($back))] != 1} {
           set ok($i,$j,$time) 1
        } elseif {[string match $datalog($i,$j,$t($time)) $datalog($i,$j,$t($back))] != 1} {
           set ok($i,$j,$time) 1
        } elseif {[string match $s_rate($i,$j,$t($time)) $s_rate($i,$j,$t($back))] != 1} {
           set ok($i,$j,$time) 1
        } else {
           set ok($i,$j,$time) 0
        }
        if {$ok($i,$j,$time) == 1} {
            set then2($i) $minus($time)
        }
     }
   }


puts "what the hell is ok in Check: $ok($i,$j,$time)"
}
proc ripple_back {} {

   global time t site who sid lat long elev noff eoff voff i Add comp sens \
        type serial theta phi freq damp coil rfile depth j new_time new_comp \
        num_sta preamp gain corner roll cv hcorner hpr new_net_flag file_cnt \
        new_file s_serial x serial_ck datalog s_rate fr_files resp_file_cnt\
        ok sign position code yr jday hr min sec file_cnt rt fieldname rfsta sitename
   global edit1 mark ck_numsta identity


  if {$edit1 == 0} {
   if {$time > 1} {
       set rbtime [expr ($time -1)]
       for {set rbtime $rbtime} {$rbtime >= 1} {incr rbtime -1} {
       set cnt 1
       set x 1
       set position($i,$j,$t($rbtime)) -
       set sid($i,$t($rbtime)) $sid($i,$t($time))
       set ok($i,$j,$rbtime) 0
puts "ok($i,$j,$rbtime) in RB is $ok($i,$j,$rbtime)"
       set identity($i,$t($rbtime)) $identity($i,$t($time))
       puts "sid sid($i,$t($rbtime)) is :$sid($i,$t($time))"
       #set new_net_flag 1
puts "we are in ripple_back!"
#puts "the select site is: $site($i,$t($time))"
```

```
        set point [array size Add]
#   if {$fieldname == "Site"} {
#      set site($i,$t($time)) $Add($fieldname)
#   }
          switch $code {

          1 {    set site($i,$t($rbtime)) $site($i,$t($time))
        puts "site($i,$t($rbtime)) is $site($i,$t($time))"
                 set sitename($i) $sitename($i)
                 set who($i,$t($rbtime)) "N"
puts "who($i,$t($rbtime)) in ripple_back is: $who($i,$t($rbtime))"
                 set lat($i,$t($rbtime)) $lat($i,$t($time))
                 set long($i,$t($rbtime)) $long($i,$t($time))
                 set elev($i,$t($rbtime)) $elev($i,$t($time))
                 set noff($i,$t($rbtime)) $noff($i,$t($time))
                 set eoff($i,$t($rbtime)) $eoff($i,$t($time))
                 set voff($i,$t($rbtime)) $voff($i,$t($time))
                 set rfsta($i,$t($rbtime)) $rfsta($i,$t($time))
                 set resp_file_cnt($i,$t($rbtime)) 0
            }
          2 {

                 set comp($i,$j,$t($rbtime)) $comp($i,$j,$t($time))
                 set type($i,$j,$t($rbtime)) $type($i,$j,$t($time))
                 set s_serial($i,$j,$t($rbtime)) $s_serial($i,$j,$t($time))
                 set rfile($i,$j,$t($rbtime)) $rfile($i,$j,$t($time))
                 set position($i,$j,$t($rbtime)) $position($i,$j,$t($time))
                 set depth($i,$j,$t($rbtime)) $depth($i,$j,$t($time))
                 set preamp($i,$j,$t($rbtime)) $preamp($i,$j,$t($time))
                 set gain($i,$j,$t($rbtime)) $gain($i,$j,$t($time))
                 set hcorner($i,$j,$t($rbtime)) $hcorner($i,$j,$t($time))
                 set hpr($i,$j,$t($rbtime)) $hpr($i,$j,$t($time))
                 set theta($i,$j,$t($rbtime)) $theta($i,$j,$t($time))
                 set phi($i,$j,$t($rbtime)) $phi($i,$j,$t($time))
                 set freq($i,$j,$t($rbtime)) $freq($i,$j,$t($time))
                 set cv($i,$j,$t($rbtime)) $cv($i,$j,$t($time))
                 set damp($i,$j,$t($rbtime)) $damp($i,$j,$t($time))
                 set coil($i,$j,$t($rbtime)) $coil($i,$j,$t($time))
                 set sens($i,$j,$t($rbtime)) $sens($i,$j,$t($time))
                 set corner($i,$j,$t($rbtime)) $corner($i,$j,$t($time))
puts "corner($i,$j,$t($rbtime)) in ripple_back is $corner($i,$j,$t($rbtime))"
                 set roll($i,$j,$t($rbtime)) $roll($i,$j,$t($time))
                 set serial($i,$j,$t($rbtime)) $serial($i,$j,$t($time))
                 set datalog($i,$j,$t($rbtime)) $datalog($i,$j,$t($time))
                 set s_rate($i,$j,$t($rbtime)) $s_rate($i,$j,$t($time))
                 set fr_files($i,$x,$t($rbtime)) $fr_files($i,$x,$t($time))
            }
          }

#ripple_up
     }
    }
   }
}

proc ripple_up {} {

   global num_sta time t site who sid lat long elev noff eoff voff i Add comp sens \
        type serial theta phi freq damp coil rfile depth j new_time new_comp \
        preamp gain corner roll cv hcorner hpr new_net_flag file_cnt \
        new_file s_serial x serial_ck datalog s_rate fr_files resp_file_cnt\
        position code yr jday hr min sec file_cnt rt fieldname rfsta sitename \
        edit1 ok mark sign ck_numsta lock chan_cnt identity
```

```
#puts " what's mark and i: $mark $i"
  if {$edit1 == 0} {
   if {$time > 1} {
      #puts "ripple_up!"
      set past [expr ($time -1)]
      #puts "what's num_sta($time) in ripple_up:  $num_sta($time)"
      #puts "what's num_sta(past) in ripple_up:  $ck_numsta($past)"
      #set prev [expr ($i - $num_sta($time))]
      set prev [expr ($num_sta($time) - $ck_numsta($past))]
      #set prev [expr ($num_sta($time) - $num_sta($past))]
      set prev [expr ($num_sta($time) - $prev)]
      #set prev [expr ($num_sta($time) - 1)]
      #puts "prev is: $prev"

      set rbtime 1
      while {$rbtime < $time} {
         #puts "the rbtime and time are: $rbtime    $time"
         for {set next $prev} {$next >= 1} {incr next -1} {
         set save $rbtime
         #if {$i == $mark} {
         #       set save $rbtime
         #       set rbtime $time
         #} else { set rbtime $save}

         set cnt 1
         set x 1
         set sid($next,$t($time)) $sid($next,$t($rbtime))
         set ok($next,$j,$time) 0
         set identity($next,$t($time)) $identity($next,$t($rbtime))
         puts "sid($next,$t($time)) in ripple_up is $sid($next,$t($time))"
         set position($i,$j,$t($rbtime)) -
         puts "the code in ripple_up is: $code"
         switch $code {

         1 {     set site($next,$t($time)) $site($next,$t($rbtime))
                 #set sitename($i) $Add($fieldname(2))
                 set who($next,$t($time)) $who($next,$t($rbtime))
#puts "who($next,$t($time)) in ripple_up is: $who($next,$t($time))"
                 set lat($next,$t($time)) $lat($next,$t($rbtime))
                 set long($next,$t($time)) $long($next,$t($rbtime))
                 set elev($next,$t($time)) $elev($next,$t($rbtime))
                 set noff($next,$t($time)) $noff($next,$t($rbtime))
                 set eoff($next,$t($time)) $eoff($next,$t($rbtime))
                 set voff($next,$t($time)) $voff($next,$t($rbtime))
                 #set rfsta($next,$t($time)) $rfsta($next,$t($rbtime))
                 set resp_file_cnt($next,$t($time)) 0
           }
         2 {
                for {set element $j} {$element <= $chan_cnt($next)} {incr element} {

                 set comp($next,$element,$t($time)) $comp($next,$element,$t($rbtime))
                 set type($next,$element,$t($time)) $type($next,$element,$t($rbtime))
                 set s_serial($next,$element,$t($time)) $s_serial($next,$element,$t($rbtime
))
                 set rfile($next,$element,$t($time)) $rfile($next,$element,$t($rbtime))
                 #set position($next,$element,$t($time)) $position($next,$element,$t($rbtim
e))
                 set depth($next,$element,$t($time)) $depth($next,$element,$t($rbtime))
                 set preamp($next,$element,$t($time)) $preamp($next,$element,$t($rbtime))
                 set gain($next,$element,$t($time)) $gain($next,$element,$t($rbtime))
                 set hcorner($next,$element,$t($time)) $hcorner($next,$element,$t($rbtime))
                 set hpr($next,$element,$t($time)) $hpr($next,$element,$t($rbtime))
                 set theta($next,$element,$t($time)) $theta($next,$element,$t($rbtime))
```

```
                    set phi($next,$element,$t($time)) $phi($next,$element,$t($rbtime))
                    set freq($next,$element,$t($time)) $freq($next,$element,$t($rbtime))
                    set cv($next,$element,$t($time)) $cv($next,$element,$t($rbtime))
                    set damp($next,$element,$t($time)) $damp($next,$element,$t($rbtime))
                    set coil($next,$element,$t($time)) $coil($next,$element,$t($rbtime))
                    set sens($next,$element,$t($time)) $sens($next,$element,$t($rbtime))
                    set corner($next,$element,$t($time)) $corner($next,$element,$t($rbtime))
                    set roll($next,$element,$t($time)) $roll($next,$element,$t($rbtime))
                    set serial($next,$element,$t($time)) $serial($next,$element,$t($rbtime))
                    set datalog($next,$element,$t($time)) $datalog($next,$element,$t($rbtime))
                    set s_rate($next,$element,$t($time)) $s_rate($next,$element,$t($rbtime))
                    set fr_files($next,$x,$t($time)) $fr_files($next,$x,$t($rbtime))
                 }
               }
           }


          }
      incr rbtime
        }
       }
      #set back 1
      #set num_sta($back) $num_sta($time)
     }
}

#proc tkerror { errmsg } {
#
# global errorInfo errorCode
#
# set msg \
#    [format "Error: %s\nResult: %s." \
#       $errmsg $errorCode]
#
#tk_dialog .dlg "Error" $msg error 0 RE-TRY
#
#}

proc check_logger {} {
   global Pf Data Value errmsg i j x t rfile time s_rate datalog fr_files stage_loop
   set x 1
   set err_code -

#   set this  [regexp {^([a-z])+[a-z0-9_-]*$} $rfile($i,$j,$t($time))]

   set datalog($i,$j,$t($time)) [string toupper $datalog($i,$j,$t($time))]
#puts "what's the data? $datalog($i,$j,$t($time))"
   #set errmsg "$datalog($i,$j,$t($time)) not in parameter file"
   if {[catch {pfgetarr Data %${Pf}#Loggers} err]} {
           error "$datalog($i,$j,$t($time)) not in parameter file"
      }
   for_array_keys data Data {
      if {$data == $datalog($i,$j,$t($time))} {
           set datalog($i,$j,$t($time)) $Data($data)
           set err_code 1
      }
   }
   if {$err_code != 1} {
         error "$datalog($i,$j,$t($time)) not in parameter file"
   }

}

proc get_fr_files {} {
    global Pf Data Value errmsg i j x t rfile time s_rate datalog fr_files stage_loop
    set check $s_rate($i,$j,$t($time))
#puts "check is: $check"
    set code -

    set datalog($i,$j,$t($time)) [string toupper $datalog($i,$j,$t($time))]
    #pfgetarr list %${Pf}#$datalog($i,$j,$t($time))#Samp
    for_array_keys data Data {
       if {$data == $datalog($i,$j,$t($time))} {
            set datalog($i,$j,$t($time)) $Data($data)
       }
    }

#puts "what's datalogger($i,$j,$t($time)): $datalog($i,$j,$t($time))"
    pfgetarr logger %${Pf}#$datalog($i,$j,$t($time))
    if {[catch {pfgetarr list %${Pf}#$datalog($i,$j,$t($time))#Samp} list_err]} {
       error "$datalog($i,$j,$t($time)) does not have a samprate of $check \
                  in parameter file"
    }
    for_array_keys lock logger {
       for_array_keys key list {
         #set Value(attribute$key) $list($key)
         #puts "what is check and key here. $check  $key"
          if {$check == $key} {
             set fr_files($i,$x,$t($time)) $list($key)
             set stage_loop $list($key)
             set code 1
#   puts " the value(attribute$key) of list($key) is:$list($key)"
          }

       }
    }
    if {$code != 1} {
         error "$datalog($i,$j,$t($time)) does not have a samprate of $check \
                  in parameter file"
    }

}

proc compute_stage {} {
    global Pf i j t time stage_loop s_rate datalog fr_files rfile site who type
    global control count rsponse_dir stage_dir net_path stage stage_cnt
    global fir_file high rsponses stages rt get_list rsponse_info rsponse_list
    global output mycount rsponse_var filters

    set rfile($i,$j,$t($time)) [string tolower $rfile($i,$j,$t($time))]
    set type($i,$j,$t($time)) [string toupper $type($i,$j,$t($time))]
    set who($i,$t($time)) [string toupper $who($i,$t($time))]
    set site($i,$t($time)) [string toupper $site($i,$t($time))]
    set rsponse $rfile($i,$j,$t($time))
    set count 2
    set high 0
    set stage_cnt 2

    pfgetarr Data %${Pf}#$datalog($i,$j,$rt($time))
    pfgetarr list %${Pf}#$datalog($i,$j,$rt($time))#Samp
    pfgetarr Sensor %${Pf}#$datalog($i,$j,$rt($time))#Channel
    pfgetarr Amps %${Pf}#$datalog($i,$j,$rt($time))#Amplifier
    pfgetarr Rate %${Pf}#$datalog($i,$j,$rt($time))#SampleRate
    set check $s_rate($i,$j,$rt($time))
    set fir_file \
         "$rfile($i,$j,$t($time))_$datalog($i,$j,$rt($time))_$s_rate($i,$j,$rt($time))"
    set rsponse_var "$rfile($i,$j,$t($time))_$datalog($i,$j,$rt($time))"
    set rsponse_info \
          "$datalog($i,$j,$rt($time))_$s_rate($i,$j,$rt($time))"
       for_array_keys rate Rate {
         if {$rate == $s_rate($i,$j,$rt($time))} {
            set high $Rate($rate)
         } else { set high $rate }
       }
    for_array_keys lock Data {
    for_array_keys key list {
       if {$check == $key} {
          set stage_loop $list($key)
         }
       }
     }
    set compare2 [lsearch $get_list $rsponse_info]
    set compare 0
    if {$i > 1} {
       set back [expr ($i - 1)]
       set compare [string match $site($i,$t($time)) $site($back,$t($time))]
    }
    set filters($i,1,$t($time)) $rfile($i,$j,$t($time))
    if {$compare == 0} {
       exec sh -c "cat $net_path/response/$rsponse > $stage_dir/$rsponse"
       if {$compare2 == -1} {
          exec sh -c "cat $net_path/response/$rsponse >> $rsponse_dir/$rsponse_info"
       }
       foreach stage $stage_loop {
          if {$stage != "NoAA"} {
            incr stage_cnt
            set filters($i,$stage_cnt,$t($time)) $stage
            exec sh -c "cat $net_path/response/$stage > $stage_dir/$stage"
            set control 1
            if {$compare2 == -1} {
               exec sh -c "cat $net_path/response/$stage >> $rsponse_dir/$rsponse_info"
            }
          }
          set stages $rsponse_var
          set rsponses $rsponse_info
       }
       lappend get_list $rsponse_info
       set stage_cnt 0
    }
       write_rsponse
}
##########################
#proc multi_stages {}
##########################
proc write_rsponse {} {
    global count net_path stgeid iunits ounits gtype decifac samprate dir dfile
    global Pf dish rsponse_info cat cat_file i j t time line one two rsponse_dir \
          defac numfield rsponse stage_info new_net s_rate datalog rsponse_var catfile
    global stage_cnt stages line_indx1 rsponses cnt get_list target rsponse_list stage_list
    global gtype ssident line_indx5 line_indx3 line_indx2 high normal ck_norm val stage_loo
p
    global mycount scount alist list2 control rt rfile fir_file
    global output
    pfgetarr class %${Pf}#Classes
    incr mycount
    #incr i -1
    #set j 1
    set normal 0
    set one 0
    set two 0
    set cnt 0
    set stage_cnt 1
    set ssident -
    set iunits -
    set ounits -
         set gtype -
    set dish "$new_net/css30/field/response"
    set dishes "$new_net/css30/field/response/$stage_info"

          set stage_file $rsponse_list
          set stage "$dish/$rsponses"
          set target "$dish/$stages"
          set classes ""
          set catfile $rsponses
    set compare [lsearch $list2 $fir_file]
    set compare2 [lsearch $rsponse_list $rsponse_var]
       if {$compare == -1} {
             lappend list2 $fir_file
          set compare2 [lsearch $rsponse_list $rsponse_var]
          set wcount 1
 #puts "rsponse_var is: $rsponse_var"
          if {$compare2 != -1} {
             lappend rsponse_list $rsponse_var
             set wcount 0
 #puts "rsponse_list is: $rsponse_list"
             foreach num $rsponse_list {
                if {$rsponse_var == $num} {
                    incr wcount
                }
             }
             set cat "$target.$wcount"
 #puts "what is target: $target.$wcount"
          } else {
             lappend rsponse_list $rsponse_var
             set cat "$target.$wcount"
 #puts "what is target2: $target.$wcount"

          }
       }
         set cnt 1
         set cat_file [open $cat a+]
               set open_id2 [open $stage r]
         for_array_keys key class {
            lappend classes $key
         }

    while {[gets $open_id2 line] >= 0} {

         set numfield [llength $line]
         set line_indx1 [lindex $line 0]
         set line_indx2 [lindex $line 1]
         set line_indx3 [lindex $line 2]
         set line_indx5 [lindex $line 4]
         if {($line_indx1 == "#")} {
               set id_flag 0
         }
 #puts $output  "classes are $classes "
 #puts $output "and line_indx2 is $line_indx2"
         if {($line_indx1 != "#") && ($numfield == 5)} {
            firstline line
               #puts "line in line_indx1 != '#' is: $line"
 #puts "cnt in write_rsponse is: $cnt"
               #puts $cat_file $line
         } elseif {([lsearch $classes $line_indx2] != -1)} {
            stage_sensor line
              stage
```

```
                incr stage_cnt
                digit
                if {$control == 1} {
                   puts $cat_file $line
                }
#               puts "classes is: $classes"
        } elseif {($line_indx2 == "Reftek") && ($numfield == 6)} {
                #puts "line is: $line"
                set linenum [llength $line]
#               puts "the line count is: $linenum"
                set gtype [lindex $line 4]
                if {$control == 1} {
                   puts $cat_file $line
                }
                set iunits -
                set ounits -
                set ssident -
                set defac -1
                set samprate -1.000000
                stage
                incr stage_cnt
                #digit
        } elseif {($numfield == 4) && ($line_indx2 == "decimal")} {
             decimate line
             stage
                incr stage_cnt
                if {$control == 1} {
                   puts $cat_file $line
                }
        } elseif {($numfield == 4) && ($line_indx3 == "normalization")} {
                #incr stage_cnt
                set ck_norm 1
                set normal $line_indx2
                #normalize line
                if {$control == 1} {
                   puts $cat_file $line
                }
        } elseif {([string length $line_indx2] == 1) && ($numfield == 2) ||
                ($numfield == 1) && ([string length $line_indx1] >= 4)} {
           secondline line
                #puts $cat_file $line
                set ck_norm 0
#                         incr cnt
        } else {
                if {$control == 1} {
                   puts $cat_file $line
                }
        }
#puts "stage_cnt is: $stage_cnt"
   }
   close $open_id2
   incr control
   #incr stage_cnt
   set stage_cnt 0
}

proc stage {} {
   global rsponse_dir paz stageid ssident defac iunits ounits gtype decifac samprate dir d
file
   global stage_dir dbout gnom rsponse stage_cnt site sens s_serial rfile yrday jday i j t
 time
   global output mycount then endtime sta chan stime ep new_net stage_time ondate filters

puts "this is how many times that we came here: $mycount"
```

```
    set decifac $defac
    set izero 0
    set gcalib 1.000000
    set leadfac 0.00000
    set stageid $stage_cnt
    #set sta [format "%-6s" $site($i,$t($time))]
    set chan [format "%-8s" $sens($i,$j,$t($time))]
    set endtime $then($i)
    #set ep [epoch "$yrday$jday"]
    #set ep $ondate
    #set ep $stage_time
    set stime $stime
    #set dir $stage_dir
    set dir "response/stage"
    if {[catch {set dfile $filters($i,$stage_cnt,$t($time))} dfile_err]} {
         puts "gottcha!"
         #puts $output "i and the rest are   $i $stage_cnt $t($time)"
    }
    if {[catch {dbaddv $dbout stage sta $sta chan $chan time $stime endtime $endtime \
                  stageid $stageid ssident $ssident gnom $gnom iunits $iunits \
                  ounits $ounits gcalib $gcalib gtype $gtype izero $izero \
                  decifac $decifac samprate $samprate leadfac $leadfac dir $dir \
                  dfile $dfile} stage_add_err]} {
                  puts "stage_out:        $stage_add_err"
         }
    set gtype -
}

proc stage_sensor { $line } {
    global Pf coil i j t time samprate s_serial ssident gtype line_indx5 iunits ounits gnom
 defac
    pfgetarr name %${Pf}#Instruments
    set gtype "sensor"
    set gnom $coil($i,$j,$t($time))
    set defac -1
    set samprate -1.000000
    set ounits "V"
    set ssident [format "%-16s" $s_serial($i,$j,$t($time))]

    for_array_keys instruments name {
       if { $line_indx5 == $instruments } {
          set iunits $name($instruments)
       }
    }
}

proc digit {} {
    global i j t time stage_cnt ssident stageid gnom iunits ounits gtype samprate dir dfile
    global rt serial defac cv high
    #incr i -1
    set ssident $serial($i,$j,$rt($time))
    set gnom 1.00000
    set iunits "V"
    set ounits "counts"
    set gtype "digitizer"
    set samprate $high
    set dir -
    set dfile -
    set defac -1
    stage
    incr stage_cnt
}

proc normalize { $line } {
   global normal line
   scan $line "%s %s %s %s" astr num norm factor

}

proc decimate { $line } {
   global ssident iunits ounits defac next high val line cat_file i j k time stage_loop
   global control gnom gtype samprate

   scan $line "%s %s %s %s" astr decimal factor value
   set gnom -
   set ssident -
   set iunits -
   set ounits -
   set gtype "FIR_decimator"
   set defac [lindex $line 3]
#   puts "high is: $high"
#   puts "i in deciamte is: $i"
   set next $high
   set val [exec sh -c "calc $high / $value"]
   #set val [exec sh -c "calc $high / $defac"]
#   puts "decimation value is $val."
   set samprate $val
   #set samprate [expr ($val * 1)]
#   puts "samprate is $samprate"
   if {$control == 1} {
      puts $cat_file [format "%s %s %s     %s" $astr $decimal $factor $value]
   }
   set high $val
   #stage
}

proc secondline {$line} {
        global Pf control line two stapar t check num_sta theo num alias \
              catfile cnt cat_file target paz seismo one cat stage_info
        global ck_norm normal next val high numfield line_indx1 one_item
         set field_cnt [expr [llength $line] -8]
         incr two
         if {($numfield == 1) && ([string length $line_indx1] >= 4)} {
            scan $line "%s" \
              hsprt
              if {$ck_norm == 1} {
              #   set normal [string tolower $normal]
                 set hsprt $normal
              } else { set hsprt $next
              }
              if {$control == 1} {
                 puts $cat_file [format "%s         " $hsprt ]
              }
         } else { scan $line "%s %s" \
              hsprt dec
              set hsprt $next
              if {$control == 1} {
                 puts $cat_file [format "%s %s         " $hsprt $dec]
              }
         }

   #set high $val
     }

proc check_rfile {} {
   global Pf Sts2 type errmsg datalog sfile check rfile i j time t

puts "this is what rfile is: $rfile($i,$j,$t($time))"
```

```
   set this  [regexp {^([a-z])+[a-z0-9_-]*$} $rfile($i,$j,$t($time))]

   #set this  [regexp {^[a-z]+[a-z0-9-]+[a-z0-9-]+[a-z0-9-_]+ \
#                [a-z0-9]*$} $rfile($i,$j,$t($time))]

#set this  [regexp {^[cC]+[mM]+[gG]+[0-9-]+[0-9tT]*$} $rfile($i,$j,$t($time))]

   puts "####################################"
   puts " this is the value of regexp: $this"

}

proc compute_rfile {} {
   global Pf Sts2 type errmsg datalog sfile check rfile i j time t

   set rfile($i,$j,$t($time)) [string tolower $rfile($i,$j,$t($time))]
   check_rfile
   set node -
   pfgetarr Class %${Pf}#Classes
   #pfgetarr responsefile %${Pf}#ResponseFile
puts "rfile is now: $rfile($i,$j,$t($time))"
   for_array_keys class Class {
      if {$rfile($i,$j,$t($time)) == $class} {
         set rfile($i,$j,$t($time)) $Class($class)
         set sfile $rfile($i,$j,$t($time))
         set check $class
         pfgetarr Sts2 %${Pf}#$sfile
         #pfgetarr Sts2 %${Pf}#$sfile
         set node 0
      }
   }
   if {$node != 0} {
         if {[catch {pfgetarr Sts2 %${Pf}#$rfile($i,$j,$t($time))} err]} {
             error "$rfile($i,$j,$t($time)) is not in parameter file"
         }
   }
   pfgetarr responsefile %${Pf}#ResponseFile
   for_array_keys quake_sensor responsefile {
      if {$sfile == $quake_sensor} {
         set try $responsefile($quake_sensor)
         set type($i,$j,$t($time)) $responsefile($quake_sensor)
         puts "the quake sensor is: $try"
         set node2 0
      }
   }
   if {$node2 != 0} {
         if {[catch {pfgetarr Sts2 %${Pf}#$rfile($i,$j,$t($time))} err]} {
             error "$rfile($i,$j,$t($time)) is not in parameter file"
         }
   }
}

proc compute_position {} {
   global i j t time theta phi position

      switch $position($i,$j,$t($time)) {
         up      {   set theta($i,$j,$t($time)) 0
                     set phi($i,$j,$t($time)) 0 }
         down    {   set theta($i,$j,$t($time)) 180
                     set phi($i,$j,$t($time)) 0 }
         north   {   set theta($i,$j,$t($time)) 90
                     set phi($i,$j,$t($time)) 0 }
         south   {   set theta($i,$j,$t($time)) 90
```

```
                        set phi($i,$j,$t($time)) 180 }
            east    {   set theta($i,$j,$t($time)) 90
                        set phi($i,$j,$t($time)) 90 }
            west    {   set theta($i,$j,$t($time)) 90
                        set phi($i,$j,$t($time)) 270 }
            default {
                        set theta($i,$j,$t($time)) 90
                        #set phi($i,$j,$t($time)) $position($i,$j,$t($time))
      if {[catch {set phi($i,$j,$t($time)) [format "%d" $position($i,$j,$t($time))]} err]} {
            error "Can only be north, south, east, west,\
                  up, down, or an integer 0 thru 360!"
        }
                    }
        }
}

proc compute_channel_codes {} {
   global Pf Sts2 errmsg cv comp coil band i j t time corner site s_rate
   global sfile Value i j x t time s_rate datalog fr_files rfile
   global  ext stage_info who s_serial rfile freq damp coil theta phi type
   global position s_serial sitename rfsta site_name rf_sta
   set rfile($i,$j,$t($time)) [string tolower $rfile($i,$j,$t($time))]
   set type($i,$j,$t($time)) [string toupper $type($i,$j,$t($time))]
   set who($i,$t($time)) [string toupper $who($i,$t($time))]
   set datalog($i,$j,$t($time)) [string toupper $datalog($i,$j,$t($time))]
   set check $s_rate($i,$j,$t($time))
   pfgetarr logger %${Pf}#Loggers
   for_array_keys data logger {
      if {$data == $datalog($i,$j,$t($time))} {
          set datalog($i,$j,$t($time)) $logger($data)
      }
   }
   pfgetarr Data %${Pf}#$datalog($i,$j,$t($time))
   pfgetarr list %${Pf}#$datalog($i,$j,$t($time))#Samp
   pfgetarr Sensor %${Pf}#$datalog($i,$j,$t($time))#Channel
   pfgetarr Amps %${Pf}#$datalog($i,$j,$t($time))#Amplifier
   pfgetarr Class %${Pf}#Classes
   pfgetarr Orient %${Pf}#Orientation
puts "site($i,$t($time)) is : $site($i,$t($time))"
   set site_name $sitename($i)
puts "sitename($i) is : $sitename($i)"
   set rf_sta $rfsta($i,$t($time))
   #puts "site_name from compute_chan is $site_name"
   set check $s_serial($i,$j,$t($time))
   set stage_info "$rfile($i,$j,$t($time)):$datalog($i,$j,$t($time)):$s_rate($i,$j,$t($tim
e))"
   set ext($stage_info) 1

   if {($who($i,$t($time)) == "YES") || ($who($i,$t($time)) == "NO") \
        || ($who($i,$t($time)) == "Y") || ($who($i,$t($time)) == "N")} {
        set up [string trimright [string range $who($i,$t($time)) 0 0]]
        set who($i,$t($time)) $up
   } else {
        puts "invalid input! plese re-enter"
   }

puts "the value of theta and phi are $theta($i,$j,$t($time)) $phi($i,$j,$t($time))"
   global Ok errorInfo
   set this  [regexp {^[a-zA-Z]+[0-9-]+[0-9]*$} $rfile($i,$j,$t($time))]
puts "this regexp: $this"

          for_array_keys Sens Sts2 {
             switch $Sens {
                response   {set rfile($i,$j,$t($time)) $Sts2($Sens)}
```

```
               freq        {set freq($i,$j,$t($time)) $Sts2($Sens)}
               damp         {set damp($i,$j,$t($time)) $Sts2($Sens)}
               const        {set coil($i,$j,$t($time)) $Sts2($Sens)}
            }
            if {$sfile == "l22_7"} {
               pfgetarr num %${Pf}#$sfile#instrument
               for_array_keys Num num {
                 puts "num is $Num"
                 if {$s_serial($i,$j,$t($time)) == $Num} {
                    set coil($i,$j,$t($time)) $num($Num)
                 }
               }
            }
            if {$sfile == "hs10"} {
               pfgetarr GenConst  %${Pf}#$sfile#gen_const
               pfgetarr Serial    %${Pf}#$sfile#Serial
               for_array_keys Const GenConst {
puts "theta($i,$j,$t($time)) is $theta($i,$j,$t($time)) and const is $Const"
                  if {$theta($i,$j,$t($time)) == $Const} {
                     set coil($i,$j,$t($time)) $GenConst($Const)
puts "coil is: $coil($i,$j,$t($time))"
                  }
               }
            }
         }

   global freq corner roll
   set info $s_rate($i,$j,$t($time))
   for_array_keys lock Data {
   for_array_keys key Sensor {
   for_array_keys key2 Amps {
      if {$key2 == "roll"} {
        set roll($i,$j,$t($time)) $Amps($key2)
      } elseif {$key2 == "$info"} {
        set corner($i,$j,$t($time)) $Amps($key2)
      } elseif {$key2 == "$info"} {
        set corner($i,$j,$t($time)) $Amps($key2)
      } else { puts " ain't none!"
      }
   }
        set Value(attribute$key) $Sensor($key)
        set ck_freq [expr (1 / $freq($i,$j,$t($time)))]
puts " the freq is: $ck_freq"
        set ck_samp $s_rate($i,$j,$t($time))
      if {$comp($i,$j,$t($time)) == $key}  {
   #puts " the value(attribute$key) of list($key) is:$Value(attribute$key)"
         set cv($i,$j,$t($time)) $Value(attribute$key)
      }
   }
   }
   set ch_code ""
############################################################
# compute the band code.
###############################

   global band
   #pfgetarr list %${Pf}#test
        if {($ck_samp >= 80.0) && ($ck_freq \
               < 10.0)} {
            append ch_code E
            set band "e"
        } elseif {($ck_samp >= 10.0) && ($ck_samp < 80.0) \
               && ($ck_freq < 10.0)} {
            append ch_code S
            set band "s"
        } elseif {($ck_samp >= 80.0) && ($ck_freq \
                >= 10.0)} {
            append ch_code H
            set band "h"
        } elseif {($ck_samp >= 10.0) && ($ck_samp < 80.0) \
                && ($ck_freq >= 10.0)} {
            append ch_code B
            set band "b"
        } elseif {($ck_samp > 1.0) && ($ck_samp < 10.0)} {
            append ch_code M
            set band "m"
        } else {
                puts "no band code for the samprate and corner!"
        }

##########################################################
# compute the instrument code
##################################

   global source rfile gain comp
        if {($comp($i,$j,$t($time)) >= 4) && ($gain($i,$j,$t($time)) < 1.0)} {
            append ch_code L
        } else {
            append ch_code H
        }

###########################################################
# compute the orientation code
###################################

   global theta phi sens comp

        set track 0
        set o 0
        set ocode($o) -
        if {($theta($i,$j,$t($time)) == 0) || ($theta($i,$j,$t($time)) == 180) \
            && ($phi($i,$j,$t($time)) == 0)} {
            #append ch_code Z
            set ocode($o) Z
        } elseif {($theta($i,$j,$t($time)) == 90) && ($phi($i,$j,$t($time)) \
            >= 350) && ($phi($i,$j,$t($time)) <= 10) || ($phi($i,$j,$t($time)) \
            >= 170) && ($phi($i,$j,$t($time)) <= 190) \
            || ($phi($i,$j,$t($time)) == 0)} {
            set ocode($o) N
        } elseif {($theta($i,$j,$t($time)) == 90) && ($phi($i,$j,$t($time)) \
           >= 80) && ($phi($i,$j,$t($time)) <= 100) || ($phi($i,$j,$t($time)) \
           >= 260) && ($phi($i,$j,$t($time)) \
           <= 280)} {
            #append ch_code E
            set ocode($o) E
        } elseif {($theta($i,$j,$t($time)) == 90) && ($phi($i,$j,$t($time)) \
            >= 315) && ($phi($i,$j,$t($time)) <= 349) || ($phi($i,$j,$t($time)) \
            >= 135) && ($phi($i,$j,$t($time)) <= 169) \
            || ($phi($i,$j,$t($time)) >= 191) && ($phi($i,$j,$t($time)) <= 224) \
            || ($phi($i,$j,$t($time)) >= 11) && ($phi($i,$j,$t($time)) <= 44)} {
            set ocode($o) 1
        } elseif {($theta($i,$j,$t($time)) == 90) && ($phi($i,$j,$t($time)) \
           >= 45) && ($phi($i,$j,$t($time)) <= 79) || ($phi($i,$j,$t($time)) \
           >= 225) && ($phi($i,$j,$t($time)) <= 269) \
           || ($phi($i,$j,$t($time)) >= 281) && ($phi($i,$j,$t($time)) <= 314) \
           || ($phi($i,$j,$t($time)) >= 226) && ($phi($i,$j,$t($time)) <= 259)} {
            set ocode($o) 2
        } else {
            puts "theta and phi values don't match code letter!"
         }
       if {$o == 0} {
          append ch_code $ocode($o)
       }

        if {$track > 0} {
          append ch_code $rcode
        }
        incr track
        incr o

   set sens($i,$j,$t($time)) $ch_code
puts "sens($i,$j,$t($time)) is $sens($i,$j,$t($time))"
       if {$sfile == "hs10"} {
            for_array_keys SerNum Serial {
puts "SerNum is $SerNum and check is $check"

               if {$SerNum == $check} {
                  foreach item $Serial($SerNum) {
puts "are we in there?"
                     set ck [lindex $item 0]
puts "ck is: $ck"
                     if {($item == 90) || \
                         ($item == 180)} {
                          set theta($i,$j,$t($time)) $item
                     }
                     if {($item == 0) || \
                         ($item >= 135) && \
                         ($item <= 225)} {
                          set phi($i,$j,$t($time)) $item
                     }
                     if {($item >= 0.90) && ($item <= 15.1)} {
                         set coil($i,$j,$t($time)) $item
                     }
                     if {($ck == "EH")} {
puts "what's the comp($i,$j,$t($time)): $comp($i,$j,$t($time))"
                         switch $comp($i,$j,$t($time)) {
                            0    {append ck Z
                                  set sens($i,$j,$t($time)) $ck
                                 }
                            1    {append item 1
                                  set sens($i,$j,$t($time)) $item
                                 }
                            2    {append item 2
                                  set sens($i,$j,$t($time)) $item
                                 }
                         }
                     }
                  }
               }
            }
         }

}

proc field_set_vals { } {

   global time t site who sid lat long elev noff eoff voff i Add comp sens \
        type serial theta phi freq damp coil rfile depth j new_time new_comp \
        preamp gain corner roll cv hcorner hpr new_net_flag file_cnt \
        new_file s_serial x serial_ck datalog s_rate fr_files resp_file_cnt \
        default_spec p
set cnt 1
set x 1
puts "can you HEAR me!!!!!!"
set new_net_flag 1
puts "are we in yet!"

      set Add($fieldname(1)) -
      set Add($fieldname(2)) -
      set Add($fieldname(3)) -999.0000
      set Add($fieldname(4)) -999.0000
      set Add($fieldname(5)) -999.0000
      set Add($fieldname(6)) 0
      set Add($fieldname(7)) 0
      set Add($fieldname(8)) 0
      set Add($fieldname(9)) 0
      set Add($fieldname(10)) -
      set Add($fieldname(11)) -1.0000000
      set Add($fieldname(12)) -
        set resp_file_cnt($i,$t($time)) $i
 select
 return $Add($fieldname($p))
}
# Show a table
#
proc set_comp_vals { } {
   global time t site who sid lat long elev noff eoff voff i Add comp sens \
        type serial theta phi freq damp coil rfile depth j new_time new_comp \
        preamp gain corner roll cv hcorner hpr new_net_flag file_cnt \
        new_file s_serial x serial_ck datalog s_rate fr_files resp_file_cnt \
        default_spec p
set cnt 1
set x 1
puts "can you HEAR me!!!!!!"
set new_net_flag 1
puts "are we in yet!"

      set Add($fieldname(1)) -
      set Add($fieldname(2)) -
      set Add($fieldname(3)) -
      set Add($fieldname(4)) 0
      set Add($fieldname(5)) 0
      set Add($fieldname(6)) 1.000000
      set Add($fieldname(7)) 0.0
      set Add($fieldname(8)) 0.0
      set Add($fieldname(9)) -
      set Add($fieldname(10)) XXX
      set Add($fieldname(13)) -
      set Add($fieldname(14)) 0
      set Add($fieldname(15)) 1.0
      set Add($fieldname(16)) 1.0
      set Add($fieldname(17)) 0.0
      set Add($fieldname(18)) 0
      set Add($fieldname(19)) 0
      set Add($fieldname(20)) 1.0
      set Add($fieldname(21)) 0
      set Add($fieldname(12)) -
        set resp_file_cnt($i,$t($time)) $i
 select
 return $Add($fieldname($p))
}
# Show a table
#
proc show_table {} {
    global Column Request First_col Wheight Wwidth Record t_flag \
        file_flag field_list file_cnt
    set w [table_init]
    set t_flag 0
    set First_col($w) 0
    #compute_rows $w
    if {$file_flag == 0} {
        defaults $w
#       fix_columns $w
#       col_create $w $First_col($w)
        #table_fill $w
    }
    if {$file_flag != 0} {
        defaults $w
#       fix_columns $w
#       by_time $w $First_col($w)
    }

    return $w
}

proc select_field {} {
        global Fields N Sequence Checkbutton
    set w [uniqueW ""]
    toplevel $w
    wm title $w $title

    set maxrow 20
    set maxfields [llength $possible]
    set lastrow [expr $maxrow+20]
    set span [expr $maxfields / $maxrow + 1]

    frame $w.f1
    button $w.f1.all \
        -text "all" \
        -command "check_all 1"
    button $w.f1.none \
        -command "check_all 0"

    pack $w.f1 -side top -fill x
    pack $w.f1.all $w.f1.none -side left -fill x -expand yes

    frame $w.f2
    button $w.f2.cancel \
        -text "cancel" \
        -command "set Ok 0 ; destroy $w"
    button $w.f2.ok \
        -text "ok" \
        -command "set Ok 1; destroy $w"
    pack $w.f2 -side bottom -fill x
    pack $w.f2.cancel $w.f2.ok -side left -fill x -expand yes

    set f $w.f3
    frame $f
    pack $f -side top -fill x

    if { [info exists Fields]} {
        unset Fields
        unset Sequence
        unset Checkbutton
        }

###############################################################
# this should be in a seperate procedure. make sure you
# do this before you write any other procedures that may
# want to use the arrange option.
##############################################################

    set i 0
    set N 1
    set expr_row $maxrow
    set expr 0

    global Possible
    set Possible $possible
    foreach field $possible {
        set Fields($field) 0
        set Sequence($field) ""
        if { [is_expression $field] } {
            set row $expr_row
            incr expr_row
            set col 0
            set expr 1
        } else {
            set row [expr $i % $maxrow + 10]
            set col [expr 2*($i / $maxrow)]
            }
        label $f.l$i -textvariable Sequence($field) -width 2
        checkbutton $f.cb$i \
            -text $field \
            -anchor w \
            -relief flat \
            -variable Fields($field) \
            -command "fix_order \{$field\}"
        set Checkbutton($field) $f.cb$i
        blt::table $f $f.l$i  $row,$col -anchor e -fill x
        incr col
        if { $expr } {
            blt::table $f $f.cb$i $row,$col -columnspan 25 -anchor w -fill x
        } else {
            blt::table $f $f.cb$i $row,$col -anchor w -fill x
            }
        incr i
        }

    global Ok
    check_current $request

    update

    grab set $w
    tkwait window $w
    grab release $w
    global Ok
    if { $Ok } {
        set wanted ""
        foreach field [array names Fields] {
            if { $Fields($field) } {
                lappend wanted $field
                }
            }
        set wanted [lsort -command by_value $wanted]
        return $wanted
    } else {
        return ""
        }
}

#########################################################
```

```
# find the default size for table
########################################################

proc compute_charsize {} {
    global Font CharWidth CharHeight LbWidth LbHeight
    listbox .l -relief ridge -width 1 -height 1
    set w0 [winfo reqwidth .l]
    set h0 [winfo reqheight .l]
    .l config -width 2 -height 2
    set w1 [winfo reqwidth .l]
    set h1 [winfo reqheight .l]
    set CharWidth [expr $w1-$w0]
    set CharHeight [expr $h1-$h0]
    set LbWidth [expr $w0-$CharWidth]
    set LbHeight [expr $w1-$CharHeight]
    destroy .l
    }


proc table_forget_columns { w } {
    set pattern $w
    append pattern {.[ml]b[0-9]*}

    foreach slave [blt::table search $w $pattern ] {
        blt::table forget $slave
        }
    }

proc fix_columns { w } {
  global Column Column_label Column_number parse t_flag file_flag
  global request First_col time

  set total 0
  set starts 0
  set time 1
  global Column_starts
  set Column_starts($w) $starts

table_forget_columns $w
  set Mapped($w) ""

  set column 2

puts "t_flag is: $t_flag"
puts "file_flag is: $file_flag"

  if {$file_flag != 0} {
     foreach one $request  {
        set name $w$one

          by_time $w $one
          set l $Column($name)
          set b $Column_label($name)

          set Column_number($name) $column
          col_map $w $b $l $column
          set column [expr $column + 1]

         }
  }
  if {$file_flag == 0} {
     foreach one $request {
        set name $w$one
```

```
            col_create $w $one
            set l $Column($name)
            set b $Column_label($name)

            set Column_number($name) $column
            col_map $w $b $l $column
            set column [expr $column + 1]

         }
    }
  }

  proc col_map { w b l col } {
        blt::table $w \
            $b 10,$col -fill x -anchor center \
            $l 11,$col -fill x -anchor center
  }

  proc col_create { w one } {
      global Column Column_label Nrows Nchars firstrow row t_flag x y
      global row many_boxes site sid comp sens rfile datalog gain s_rate i j k time t \
                  num_sta chan_cnt parse l counter type cat locate z field \
                  sp_fields default_spec
      set locate ""
     set z 1
      set n [uniqueN]
      set name $w$one
      set nchars 10
      set nrows 20
      set counter 0
      set x 1
      set y 1

      lappend locate $name
      set cat [string trimleft [string range $locate 3 end]]
                     set tail($x,$y,$t($time)) [field_values $one $x]

      set b $w.mb$n

      menubutton $b -relief raised -menu $b.m -text $one
      set q 0
      set l $w.lb$n
      lappend many_boxes $l
      puts "manyboxes is: $many_boxes"

      listbox $l -relief ridge -width ${parse} -height ${nrows} \
      -yscrollcommand "$w.row  set"
      bind $l <Any-ButtonRelease> "set_entry %W $name"
      #bind $l <Any-ButtonRelease>

      menu $b.m
          #       $b.m add separator
                  $b.m add command -label "Delete column" -command "del_col $name $i"
                  $b.m add command -label "Sort"          -command "uc"
                  $b.m add command -label "Subset"        -command "uc"
  puts " numsta is : $num_sta($time) in col_create"
      for {set x 1} {$x <= $num_sta($time)} {incr x} {
          for {set y 1} {$y <= $chan_cnt($x)} {incr y} {

                     set tail($x,$y,$t($time)) [field_values $one $x]
                     #set sp_fields($one) [field_values $one]

                     if {$y > $chan_cnt($x)} {set y 1}
                     if {$one == "filterresponsefiles"} {
                            compute_fr_files
                            $1 insert end $field
                        } else { $1 insert end $tail($x,$y,$t($time)) }
                       incr counter
            }
        }
#          show_record $w
#          table_fill $w
#          if {$t_flag > 0} {
#                 table_fill $w $1
#          }

        global Column_label Column Parent Expression Name
        set Column_label($name) $b
        set Column($name) $1
        set Parent($name) $w
        set Expression($name) $one
        set Name($1) $name
        }
      proc del { w 1 } {
           global Record t_flag column

           if {($t_flag > 0) && ($Record($w) != 0)} {
              $1 delete 0 [expr ($Record($w) - 1)]
           }

      }

######################################################
#procedure to output correct field values for each
#stapar.
######################################################
proc field_values { one x } {

      global tail  y t time site who sid sens comp rfile s_rate datalog \
           gain preamp corner roll cv hcorner hpr lat long elev noff \
           eoff voff type serial theta phi coil freq damp depth\
           fr_files z resp_file_cnt parse new default_spec s_serial\
           epoch_clk serial_ck clock rt new_net_flag

      global Site Who Comp DSerial SSerial SampleRate DataLogger Damp \
           Coil dbPreAmp dbGain Roll CountsV Highcorner HPR Freq Corner



                                    if {$one == "station"} {
                            set parse [expr ([string length $one] + 2)]
                            set tail($x,$y,$t($time)) $site($x,$t($time))
                       } elseif {$one == "sid"} {
                            set parse [expr ([string length $one] + 2)]
                            set tail($x,$y,$t($time)) $sid($x,$t($time))
                       } elseif {$one == "is_it_running"} {
                            set parse [expr ([string length $one] + 2)]
                            set tail($x,$y,$t($time)) $who($x,$t($time))
                       } elseif {$one == "sens"} {
                            set parse [expr ([string length $one] + 2)]
                            set tail($x,$y,$t($time)) $sens($x,$y,$t($time))
                       } elseif {$one == "component"} {
                            set parse [expr ([string length $one] + 2)]
                            set tail($x,$y,$t($time)) $comp($x,$y,$t($time))
                       } elseif {$one == "dserial"} {
                               set parse [expr ([string length $one] + 2)]
                               set tail($x,$y,$t($time)) $serial($x,$y,$t($time))
                       } elseif {$one == "rsponsefile"} {
```

```
            set parse [expr ([string length $one] + 8)]
            set tail($x,$y,$t($time)) $rfile($x,$y,$t($time))
        } elseif {$one == "samplerate"} {
            set parse [expr ([string length $one] + 8)]
            set tail($x,$y,$t($time)) $s_rate($x,$y,$t($time))
        } elseif {$one == "datalogger"} {
            set parse [expr ([string length $one] + 8)]
            set tail($x,$y,$t($time)) $datalog($x,1,$t($time))
        } elseif {$one == "gain"} {
            set parse [expr ([string length $one] + 4)]
            set tail($x,$y,$t($time)) $gain($x,$y,$t($time))
        } elseif {$one == "preamp"} {
            set parse [expr ([string length $one] + 1)]
            set tail($x,$y,$t($time)) $preamp($x,$y,$t($time))
        } elseif {$one == "hf_corner"} {
            set parse [expr ([string length $one] + 1)]
            set tail($x,$y,$t($time)) $corner($x,$y,$t($time))
        } elseif {$one == "hf_roll"} {
            set parse [expr ([string length $one] + 1)]
            set tail($x,$y,$t($time)) $roll($x,$y,$t($time))
        } elseif {$one == "digit_const"} {
            set parse [expr ([string length $one] + 1)]
            set tail($x,$y,$t($time)) $cv($x,$y,$t($time))
        } elseif {$one == "hp_corner"} {
            set parse [expr ([string length $one] + 1)]
            set tail($x,$y,$t($time)) $hcorner($x,$y,$t($time))
        } elseif {$one == "hp_roll"} {
            set parse [expr ([string length $one] + 1)]
            set tail($x,$y,$t($time)) $hpr($x,$y,$t($time))
        } elseif {$one == "Time"} {
            set parse [expr ([string length $one] + 25)]
            #set tail($x,$y,$t($time)) $rt($time)
            set tail($x,$y,$t($time)) $epoch_clk($time)
            #set tail($x,$y,$t($time)) $clock
            #set tail($x,$y,$t($time)) $new
        } elseif {$one == "EndTime"} {
            set parse [expr ([string length $one] + 8)]
            set tail($x,$y,$t($time)) $clock
            #set tail($x,$y,$t($time)) $new
        } elseif {$one == "latitude"} {
            set parse [expr ([string length $one] + 10)]
            set tail($x,$t($time)) $lat($x,$t($time))
        } elseif {$one == "longitude"} {
            set parse [expr ([string length $one] + 10)]
            set tail($x,$t($time)) $long($x,$t($time))
        } elseif {$one == "elevation"} {
            set parse [expr ([string length $one] + 8)]
            set tail($x,$t($time)) $elev($x,$t($time))
        } elseif {$one == "n_offset"} {
            set parse [expr ([string length $one] + 8)]
            set tail($x,$t($time)) $noff($x,$t($time))
        } elseif {$one == "e_offset"} {
            set parse [expr ([string length $one] + 8)]
            set tail($x,$t($time)) $eoff($x,$t($time))
        } elseif {$one == "v_offset"} {
            set parse [expr ([string length $one] + 8)]
            set tail($x,$t($time)) $voff($x,$t($time))
        } elseif {$one == "type"} {
            set parse [expr ([string length $one] + 2)]
            set tail($x,$y,$t($time)) $type($x,$y,$t($time))
        } elseif {$one == "sserial"} {
            set parse [expr ([string length $one] + 2)]
            set tail($x,$y,$t($time)) $s_serial($x,$y,$t($time))
        } elseif {$one == "theta"} {
```

```
                        set parse [expr ([string length $one] + 2)]
                        set tail($x,$y,$t($time)) $theta($x,$y,$t($time))
                    } elseif {$one == "phi"} {
                        set parse [expr ([string length $one] + 2)]
                        set tail($x,$y,$t($time)) $phi($x,$y,$t($time))
                    } elseif {$one == "freq"} {
                        set parse [expr ([string length $one] + 8)]
                        set tail($x,$y,$t($time)) $freq($x,$y,$t($time))
                    } elseif {$one == "damping"} {
                        set parse [expr ([string length $one] + 8)]
                        set tail($x,$y,$t($time)) $damp($x,$y,$t($time))
                    } elseif {$one == "gen_const"} {
                        set parse [expr ([string length $one] + 8)]
                        set tail($x,$y,$t($time)) $coil($x,$y,$t($time))
                    } elseif {$one == "depth"} {
                        set parse [expr ([string length $one] + 1)]
                        set tail($x,$y,$t($time)) $depth($x,$y,$t($time))
                    } elseif {$one == "filterresponsefiles"} {
                        set parse [expr ([string length $one] + 43)]
                    } else {
                        set parse [expr ([string length $one] + 1)]
                        set tail($x,$y,$t($time)) $type($x,$y,$t($time))
                    }

}

proc compute_fr_files {} {
   global tail one fr_files x y t time resp_file_cnt l field

   set field ""
   switch $resp_file_cnt($x,$t($time)) {
        1    {set tail($x,1,$t($time)) $fr_files($x,1,$t($time))
              set z 1
                lappend field $tail($x,$z,$t($time))
             }
        2    {
                set tail($x,1,$t($time)) $fr_files($x,1,$t($time))
                set tail($x,2,$t($time)) $fr_files($x,2,$t($time))
                set z 1
                for {set z 1} {$z <= 2} {incr z} {
                        lappend field $tail($x,$z,$t($time))
                }
             }
        3    {
                set tail($x,1,$t($time)) $fr_files($x,1,$t($time))
                set tail($x,2,$t($time)) $fr_files($x,2,$t($time))
                set tail($x,3,$t($time)) $fr_files($x,3,$t($time))
                set z 1
                for {set z 1} {$z <= 3} {incr z} {
                        lappend field $tail($x,$z,$t($time))
                }
             }
        4    {
                set tail($x,1,$t($time)) $fr_files($x,1,$t($time))
                set tail($x,2,$t($time)) $fr_files($x,2,$t($time))
                set tail($x,3,$t($time)) $fr_files($x,3,$t($time))
                set tail($x,4,$t($time)) $fr_files($x,4,$t($time))
                set z 1
                for {set z 1} {$z <= 4} {incr z} {
                        lappend field $tail($x,$z,$t($time))
                }
             }
        5    {
                set tail($x,1,$t($time)) $fr_files($x,1,$t($time))
        set tail($x,2,$t($time)) $fr_files($x,2,$t($time))
        set tail($x,3,$t($time)) $fr_files($x,3,$t($time))
        set tail($x,4,$t($time)) $fr_files($x,4,$t($time))
        set tail($x,5,$t($time)) $fr_files($x,5,$t($time))
        set z 1
        for {set z 1} {$z <= 5} {incr z} {
                lappend field $tail($x,$z,$t($time))
        }
    }
6   {
        set tail($x,1,$t($time)) $fr_files($x,1,$t($time))
        set tail($x,2,$t($time)) $fr_files($x,2,$t($time))
        set tail($x,3,$t($time)) $fr_files($x,3,$t($time))
        set tail($x,4,$t($time)) $fr_files($x,4,$t($time))
        set tail($x,5,$t($time)) $fr_files($x,5,$t($time))
        set tail($x,6,$t($time)) $fr_files($x,6,$t($time))
        set z 1
        for {set z 1} {$z <= 6} {incr z} {
                lappend field $tail($x,$z,$t($time))
        }
    }
7   {
        set tail($x,1,$t($time)) $fr_files($x,1,$t($time))
        set tail($x,2,$t($time)) $fr_files($x,2,$t($time))
        set tail($x,3,$t($time)) $fr_files($x,3,$t($time))
        set tail($x,4,$t($time)) $fr_files($x,4,$t($time))
        set tail($x,5,$t($time)) $fr_files($x,5,$t($time))
        set tail($x,6,$t($time)) $fr_files($x,6,$t($time))
        set tail($x,7,$t($time)) $fr_files($x,7,$t($time))
        set z 1
        for {set z 1} {$z <= 7} {incr z} {
                lappend field $tail($x,$z,$t($time))
        }
    }
8   {
        set tail($x,1,$t($time)) $fr_files($x,1,$t($time))
        set tail($x,2,$t($time)) $fr_files($x,2,$t($time))
        set tail($x,3,$t($time)) $fr_files($x,3,$t($time))
        set tail($x,4,$t($time)) $fr_files($x,4,$t($time))
        set tail($x,5,$t($time)) $fr_files($x,5,$t($time))
        set tail($x,6,$t($time)) $fr_files($x,6,$t($time))
        set tail($x,7,$t($time)) $fr_files($x,7,$t($time))
        set tail($x,8,$t($time)) $fr_files($x,8,$t($time))
        set z 1
        for {set z 1} {$z <= 8} {incr z} {
                lappend field $tail($x,$z,$t($time))
        }
    }
9   {
        set tail($x,1,$t($time)) $fr_files($x,1,$t($time))
        set tail($x,2,$t($time)) $fr_files($x,2,$t($time))
        set tail($x,3,$t($time)) $fr_files($x,3,$t($time))
        set tail($x,4,$t($time)) $fr_files($x,4,$t($time))
        set tail($x,5,$t($time)) $fr_files($x,5,$t($time))
        set tail($x,6,$t($time)) $fr_files($x,6,$t($time))
        set tail($x,7,$t($time)) $fr_files($x,7,$t($time))
        set tail($x,8,$t($time)) $fr_files($x,8,$t($time))
        set tail($x,9,$t($time)) $fr_files($x,9,$t($time))
        set tail($x,10,$t($time)) $fr_files($x,10,$t($time))
        set z 1
        for {set z 1} {$z <= 9} {incr z} {
                lappend field $tail($x,$z,$t($time))
        }
    }
```

```
        }
    }


    ######################################################
    #procedure to output data from all stapars in one
    #network.
    ######################################################
    proc by_time { w one } {
            global many_boxes file pattern new stapar time new_file \
                    i j k t time site comp rt type serial theta \
                    field phi freq damp coil rfile sens depth y
            global Column Column_label Nrows Nchars firstrow row
            global sid comp sens who rt hold lookup \
                    num_sta chan_cnt parse l counter file_cnt t_flag
            set n [uniqueN]
            set name $w$one
            #set time 1
            set nchars 10
            set nrows 20
            set counter 0
            set x 1
            set y 1

            set b $w.mb$n

               set time 1
            menubutton $b -relief raised -menu $b.m -text $one

            set test [field_values $one $x]
            #set tail($x,$y,$t($time)) [field_values $one $x]
            set l $w.lb$n
            lappend many_boxes $l

            listbox $l -relief ridge -width ${parse} -height ${nrows} \
            -yscrollcommand "$w.row  set"
            bind $l <Any-ButtonRelease> "set_entry %W $name"
            #bind $l <Any-ButtonRelease>

            menu $b.m
                    $b.m add separator
                    $b.m add command -label "Delete column" -command "del_col $name $i"
                    $b.m add command -label "Sort"          -command "uc"
                    $b.m add command -label "Subset"        -command "uc"
    puts "file_cnt at this point is: $file_cnt"
    puts "num_sta($time) at this point is: $num_sta($time)"
               while {$time <= [expr ($file_cnt - 1)]} {
            for {set x 1} {$x <= $num_sta($time)} {incr x} {
                  for {set y 1} {$y <= $chan_cnt($x)} {incr y} {
                       set tail($x,$y,$t($time)) [field_values $one $x]
                       if {$y > $chan_cnt($x)} {set y 1}
                       if {$one == "filterresponsefiles"} {
                            compute_fr_files
                            $l insert end $field
                        } else { $l insert end $tail($x,$y,$t($time)) }
                       incr counter
                    }
              }
                    incr time
              #incr time -1
            }

    #       if {$t_flag > 0} {
    #               table_fill $w $l
```

```
#       }

        global Column_label Column Parent Expression Name
        set Column_label($name) $b
        set Column($name) $1
        set Parent($name) $w
        set Expression($name) $one
        set Name($1) $name
    }

#############################################################
#Procedure will create listbox with a scrollbar in order to
#view the files for a certain network. The files will be
#listed in the form of buttons. Click on button to open the
#stapar file.
#############################################################

proc search {entry } {
   global key pattern new clock
   set s .search
   toplevel $s
   set name $pattern
   wm title $s $name
   label $s.space
   frame $s.bar -relief raised -bd 2
   frame $s.dummy
   pack $s.bar $s.dummy -side top -fill x

   menubutton $s.bar.view -text View \
                 -menu $s.bar.view.menu
   menubutton $s.bar.help -text Help \
                 -menu $s.bar.help.menu
   pack $s.bar.view -side left
   pack $s.bar.help -side right

   menu $s.bar.view.menu
   $s.bar.view.menu add command \
                 -label "by stapar" -command "uc"
   $s.bar.view.menu add command \
                 -label "by station" -command "all"
   menu $s.bar.help.menu
   $s.bar.help.menu add command \
                 -label "under construction" -command "uc"

   button $s.dismiss -text Dismiss -command "destroy $s"
   pack $s.dismiss -side bottom -fill x
   set files [exec sh -c "ls $pattern/*0S"]
   set key $files
   set i 1
   foreach stapar $files {
        set new [file tail $stapar]
        set file_flag 1
        set clock $new
        button $s.run($i) \
                 -text $new\
                 -anchor s \
                 -relief ridge \
                 -command "set clock $new; \
                 pre_start "

        incr i
   }
   incr i -1
   set j 1
```

```
    set k 1
    set l 1
    set  maxcol 8
    set up 0
    set col [expr $up % $maxcol]
    set row [expr $up / $maxcol + 10 ]
    set step [expr $row + 1]

    for {set y 1} {$y <= $i} {incr y} {
         set col [expr $up % $maxcol]
         set row [expr $up / $maxcol + 10 ]
         blt::table $s.dummy $s.run($y) $row,$col \
                 -anchor center -fill x
         incr up
    }
    focus $s.bar
}

###################################################
#Procedure to produce a text window about field2db
#Creates a menu in order to manipulate stapar files.
#Options to create new, open, and close for now.
#There will be a help menu to describe what all buttons
#are for.
########################################################
proc box {} {
        global Pf argv0 net_path is_net yr jday hr min sec j file_cnt \
             stacode piece f time_cnt id_cnt env counter time new_net_flag
        tk_bisque

    lookNlogs
    regsub {.*/} $argv0 "" argv0
    set new_net_flag 0
    set f [uniqueF]
    set stacode 1
    set time 0
    set time_cnt 1
    set counter 0
    set file_cnt 0
    set j 1
    set yr 0
    set jday 0
    set hr 0
    set min 0
    set sec 0
    set id_cnt 1
    regsub {.*/} $argv0 "" argv0
    set piece ""
    set Pf .$argv0

         set env(NET_PATH) /hf/hifreq
         set net_path $env(NET_PATH)
         set Nrows_default [pfget $Pf jora]
         set net [pfgetlist @$Pf#Networks]
         set t .top

         label .space
         frame .mbar -relief raised -bd 2
         frame .dummy -width 10c -height 5m
         pack .mbar .dummy -side top -fill x
         menubutton .mbar.file -text File -menu .mbar.file.menu
         menubutton .mbar.help -text Help -menu .mbar.help.menu
         pack .mbar.file -side left
         pack .mbar.help -side right
        menu .mbar.file.menu
        .mbar.file.menu add cascade -label "Open" \
                        -menu .mbar.file.menu.open
        menu .mbar.file.menu.open
        foreach x $net {
              .mbar.file.menu.open add command -label $x \
                        -command "net_search $x"
        }

        .mbar.file.menu add command -label "New" \
                        -command "new_network $f"
        .mbar.file.menu add cascade -label "Edit" \
                        -menu .mbar.file.menu.edit
        menu .mbar.file.menu.edit
        foreach y $net {
              .mbar.file.menu.edit add command -label $y \
                        -command "FindDb $y"
        }
        .mbar.file.menu add command -label "Quit" \
                        -command {puts stdout \
        "Bye! "; exit 0}
        menu .mbar.help.menu
        .mbar.help.menu add command -label "Under Construction!" \
                        -command "uc"

        tk_menuBar .mbar .mbar.file  .mbar.help \
        focus .mbar

        label .program \
                -text "Field_To_Db" \
                -font -*-helvetica-bold-r-*-*-*-240-*-*-*-*-*-*
        label .version \
                -text "Version 1.0" \
                -font -*-helvetica-bold-r-*-*-*-140-*-*-*-*-*-*
        label .blank\
                -text "" \
                -font -*-helvetica-bold-r-*-*-*-140-*-*-*-*-*-*


        button .quit -text Quit -command "exit 0"
        pack .program .version .space .quit \
                 -anchor w -fill x
puts " jora says: $Nrows_default"

}
set Wcnt 0
set Fcnt 0
set many_boxes ""
set rsponse_list ""
set instr_match ""
set insname_match ""
set name_match ""
set get_list ""
set Maxheight 20
set Maxwidth 20
set firstrow 0
tk_bisque
set new_net_flag 0
set time 0
set file_cnt 0
set inid 1
set j 1
set yr 0
set jday 0
set hr 0
set min 0
set sec 0
set lock 0
set site_cnt 1
set speed_ck 0
set stage_cnt 2
set stage_info 0
set scount 1
set list_cnt 2
set sign "OFF"
set mark 1
set loop 0
set list2 ""
set sublist ""
set tmlist ""
set alist ""
set stage_info ""
set control 1
set Site "station "
set Who "is it running? (y/n)"
set Latitude "latitude (decimal degrees)"
set Longitude "longitude (decimal degrees)"
set Elevation "elevation (km)"
set N_offset "n-offset (km)"
set E_offset "e-offset (km)"
set V_offset "v-offset (km)"
set DSerial "data logger serial number"
set DataLogger "data logger type"
set SampleRate "sample rate (s/s)"
set Model "sensor model"
set Comp "component number"
set Depth "emplacement depth (km)"
set Theta "verticle orientation"
set Phi   "horizontal orientation"
set SSerial "sensor serial number"
set Freq "low freq corner (Hz)"
set Damp "damping"
set Coil "generator constant"
set dbPreAmp "pre-amp gain (db)"
set dbGain "gain (db)"
set Corner "high freq corner (Hz)"
set Roll "high freq roll-off"
set CountsV "digitizer constant (cnts/v)"
set Highcorner "high pass filter corner (Hz)"
set HPR "high pass filter roll-off"
   set high 128000.0000
set mycount 0
set site_go 0
set edit1 0
set insname_loop 0
box
```